Systematic Message Schedule Construction for Time-Triggered CAN

Klaus Schmidt and Ece G. Schmidt

***Abstract*—The most widely used standard for in-vehicle communication networks that interconnect electronic control units is the controller area network (CAN). However, the event-triggered architecture of CAN introduces several issues, such as predictability, signal jitter, and reliability. Different time-triggered networks are being developed to address these issues. In this paper, we focus on time-triggered CAN (TTCAN), which achieves time-triggered behavior by implementing time-division multiple access on the existing CAN network standard. The main task is thus to construct a message schedule for a given set of messages while fulfilling certain performance criteria. To this end, we provide a formal framework for the construction of feasible message schedules in TTCAN networks by considering several performance metrics, such as bandwidth utilization and jitter, as well as the hardware constraints of the TTCAN controller specification.**

***Index Terms*—Performance analysis, real time, scheduling, time-triggered controller area network (TTCAN), vehicular communication networks.**

I. Introduction

In-VEHICLE electronic systems have been replacing their mechanical and hydraulic counterparts since the 1970s. The growing capabilities and reliability of hardware and software enable to add new functionalities. In today's vehicles, some electronic systems, such as the antilock braking system, electronic stability program, and electric power steering, are responsible for driving safety and comfort. Furthermore, devices in the vehicle body such as windows, wipers, and communication and entertainment equipment, such as the radio, digital video disc, and navigation systems, are also controlled by *electronic control units* (ECUs). ECUs are embedded systems with a microcontroller, sensors, and actuators, and they require communication to execute most of their tasks. In today's luxury cars, up to 70 ECUs exchange 2500 signals using networks that are similar to computer networks [1], [2].

Communication networks in vehicles or, in general, embedded systems require efficient and timely operation. Most of the messages transmitted over these networks are *sporadic* or *periodic* real-time messages that have *deadlines* defining the latest transmission time. Sporadic messages are generated when certain events, such as the braking action of a driver, occur. However, the sporadic messages also have periods that determine the minimum time interval between any two consecutive generations of these messages.

In-vehicle communication networks can be classified as event-triggered, time-triggered, and hybrid networks according to methods of granting medium access to the ECU nodes. In *event-triggered* communication, the messages between ECU nodes are generated based on the occurrence of significant events, and network access is dynamically granted to ECUs based on message priorities. *Time-triggered* communication controls the medium access using predefined time windows [time-division multiple access (TDMA)]. The features of these two access methods are combined in *hybrid medium access*. A typical approach is dividing the time into periods allocated for real-time and nonreal-time traffic and allocating fractions of bandwidth for both types of traffic [1], [2].

Different network architectures were developed to perform different types of functions at different speeds. The Local Interconnect Network [3] is a low-speed/low-cost network designed for communication between ECUs and their sensors and actuators. It supports applications such as controlling of doors, seats, windows, or climate. Faster networks (up to and above 1 Mb/s), such as Time-Triggered Protocol (TTP) [4], ByteFlight [5], or FlexRay [6], are used for applications that require robust and predictable operation such as x-by-wire systems. These networks are all time triggered or hybrid [2].

The most widely used in-vehicle communication network in automotive systems is the controller area network (CAN) [2], [7]. It provides low-cost robust communication with bounded network delay among ECUs and sensors with a data rates between 125 kb/s and 1 Mb/s and is used for a very wide range of applications, such as chassis control and power train. CAN is an event-triggered network. The nodes access the network based on the priority of the frame to be transmitted.

The new functionalities in automotive electronics such as x-by-wire systems require deterministic behavior and high performance. Although the event-triggered CAN bus can provide bounded delay for medium access, it is not possible to know the exact time instant when a given message will be sent in advance, which leads to jitter of periodic messages. In addition, if there are transmission errors, the automatic retransmission feature of the CAN bus might increase the load, and the messages can miss their deadlines [8].

These issues motivate the use of a time-triggered approach for today's automotive technologies. To this end, several time-triggered networks, such as TTP, FlexRay, ByteFlight, and time-triggered CAN (TTCAN), have been recently developed [9], [10]. In this paper, we focus on the TTCAN protocol that

Manuscript received February 15, 2007; revised May 30, 2007 and July 17, 2007. The review of this paper was coordinated by Prof. X. Shen.

K. Schmidt is with the Institute of Control and Automation, University of Erlangen-Nuremberg, 91058 Erlangen, Germany (e-mail: klaus.schmidt@rt.eei.uni-erlangen.de).

E. G. Schmidt is with the Department of Electrical and Electronics Engineering, Middle East Technical University, Ankara 06531, Turkey (e-mail: eguran@metu.edu.tr).

Digital Object Identifier 10.1109/TVT.2007.906413

imposes a TDMA structure over the CAN medium access according to the stable International Standards Organization standard as of 2004 [11]. Messages are sent in a time-triggered fashion, and conflicts are resolved by the bit-wise arbitration mechanism of CAN. TTCAN can be implemented using the existing CAN network infrastructure and with small and inexpensive changes to the current CAN chips. Hence, economical gradual migration from CAN to TTCAN is possible to provide deterministic behavior and better real-time performance [7], [8]. A detailed comparison of the performance of CAN and TTCAN networks is provided in [1]. It is stated that TTCAN networks have less average latency response time and jitter compared to event-triggered CAN when reacting to asynchronous external events.

In this paper, we address the message-scheduling problem in TTCAN networks. We provide a formal description of the problem including the definition of performance metrics such as bandwidth utilization and jitter. Analyzing the constraints that result from the TTCAN specifications and the hardware implementation, we identify properties of message periods that are favorable for the scheduling process. Based on this result, we present a systematic approach to construct feasible message schedules that guarantee that all messages meet their deadlines for both periodic and sporadic messages. Additionally, our approach takes the performance metrics into consideration to produce efficient message schedules.

The structure of this paper is as follows. In Section II, the TTCAN protocol, constraints, performance metrics, and scheduling issues are discussed. The impact of the message properties on the performance of the TTCAN network is analyzed in Section III. Our systematic approach for message scheduling is presented in Section IV. Conclusions and an outline of future work are given in Section V.

II. TTCAN Protocol

A. Description

The TTCAN network is designed as another layer that operates on the event-triggered CAN network. It uses standard CAN frames and the CAN medium access technique in addition to its time-triggered access. Hence, we first describe the features of the standard CAN protocol that are relevant to our work.

1) Event-Triggered CAN: The signals to be transmitted on the CAN network are packed in frames with unique identifiers. The network access is based on the frame priorities carried in the frame identifier. CAN2.0A (or "standard CAN"), which is used for in-vehicle communications, has an 11-bit identifier.

The nonreturn-to-zero bit representation with bit stuffing of length 5 is used in CAN frames to maintain bit timing. The standard CAN data frame can contain 0–8 data bytes, and the largest CAN frame is 135 bits, including all the protocol overheads. More precisely, for a signal j with b_j data bits, and thus $d_j = \lceil b_j/8 \rceil$ data bytes, a message of length

$$C_j = 47 + 8d_j + \left\lfloor \frac{34 + 8d_j}{4} \right\rfloor \quad (1)$$

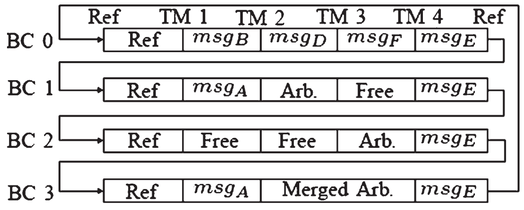


Fig. 1. TTCAN SM.

is constructed ($\lceil\cdot\rceil$ and $\lfloor\cdot\rfloor$ denote the ceiling and floor operators, respectively). In this expression, 47 is the size of the fixed-form bit fields of the CAN frame, and $\lfloor(34 + 8d_j/4)\rfloor$ is the amount of the required bit stuffing, where stuffed bits are also subject to bit stuffing [12].

The medium access is given by a carrier-sense multiple-access scheme with a nondestructive bit-wise arbitration protocol. Any CAN node may start a transmission when the bus is idle. If at least two nodes start to transmit at the same time, the conflict is resolved with a bit-wise arbitration of the frame identifiers by wired-and-mechanism. The dominant bits overwrite the recessive bits, and the stations that observe a different bit on the bus than the bit that they sent stop transmission and start listening. Hence, the identifier with the lowest binary value has the highest priority. The node that loses the arbitration retransmits its frame when the bus becomes idle. The arbitration mechanism relies on the fact that all of the messages have distinct identifiers (11 bits for the base frame format). If a transmission error occurs on a CAN network, then the corrupted frame automatically reenters the next arbitration phase [2], [7]. Medium access delay bounds can be computed if the signal periods and the minimum interarrival times of the sporadic signals are known [13].

2) TTCAN: The time-triggered behavior of the TTCAN network is achieved by implementing synchronous timing in each network node. A *reference message* is periodically transmitted to ensure that all nodes are synchronized to a common global time. After each reference message, the nodes increment a local counter once every *network time unit* (NTU). NTU is the unit of the *cycle time* in the TTCAN network.

There are two levels of synchronization in TTCAN networks. Level 1 synchronization provides the minimum precision required to support the time-triggered scheduling. The NTU in Level 1 is the network bit time. Level 2 synchronization, which can provide a high-precision global time base related to the physical second, makes it possible to synchronize and interface TTCAN to other networks such as TTP.

The time-triggered operation is achieved by using a fixed precomputed schedule that cyclically repeats. The schedule is organized as a matrix called the *system matrix* (SM; see Fig. 1). The rows and columns of the SM form time windows. The beginning of each time window [each transmission column (TC)] is indicated by a *time mark* (TM). Each row of the SM is called a *basic cycle* (BC) that starts with a reference message transmitted by a node in the network that assumes the role of the time master. The reference message carries 1 B of data for Level 1 CAN realization and 4 B of data for Level 2 CAN realization. In this paper, we assume that the reference message

TABLE I
TRIGGERS FOR NODE n

Trigger	Window Type	Pointer	TC	CO	RF
Tx_Trigger 1	Exclusive	msg_A	TM 1	1	2
Tx_Trigger 2	Arbitrating	msg_K	TM 3	2	0
Rx_Trigger 1	Exclusive	msg_D	TM 2	0	0

is 4 B, and when it is packed, it generates a 95-bit frame [see (1)]. The time windows in the SM have three different types. An *exclusive* time window is assigned to a single specific message. No other messages are transmitted during an exclusive window. An *arbitrating* time window is assigned to several messages that are transmitted by a group of nodes. Possible conflicts among these messages are resolved by the bit-wise arbitration mechanism of CAN. Messages are transmitted in the arbitrating window only if there is sufficient time remaining to complete the message transmission. Two or more arbitrating windows that appear back to back can be merged. The *free* time windows have no scheduled messages and are reserved for future use. The retransmission functionality of the event-triggered CAN is disabled in TTCAN, except for unsuccessful reference messages and in merged arbitrating windows.

A node in TTCAN stores the information for the TMs of its own messages in register sets called *triggers*. The *transmit triggers* (Tx_Trigger) store information related to the sent messages, and the *receive triggers* (Rx_Trigger) store information related only to the messages to be received in exclusive windows. Additionally, the reference message has its own trigger (Ref_Trigger). Once a Tx_Trigger is activated, the node can start transmitting during a maximum time interval known as the Tx_Enable window. Each trigger has a pointer to the related message, a reference to the TC that the related message is transmitted, and an indication of the type of window. A message can be periodically repeated multiple times in a TC. In this case, the trigger contains a Cycle_Offset (CO) that gives the BC in which the message appears the first time in TC. The period of the message repeated in the TC is stored in the Repeat_Factor (RF) field, which is set to 0 if the message does not periodically appear [9].

To illustrate the above concepts, we consider a given node n on the TTCAN network that sends a periodic message msg_A and a sporadic message msg_K and receives a periodic message msg_D. An example SM is depicted in Fig. 1, and the corresponding trigger information is shown in Table I.

The SM is column oriented. Once a column size is decided for the first BC, it has to stay constant. The types of windows in the same column can be different for different BCs.

B. TTCAN Scheduling: Issues and Previous Work

The transmission schedule for the TTCAN SM is static, and it is computed offline. There are several constraints and performance metrics to be considered when the SM is computed.

- *Schedulability constraint:* The real-time messages must be transmitted before their deadlines. This is very important, especially considering the safety critical nature of automotive applications.
- *Hardware constraints:* The triggers that contain information about the sent and received messages are stored in a fixed-size memory on the TTCAN chip. Hence, the total number of triggers per node is limited.
- *Utilization:* This metric defines how efficiently the network bandwidth is used. Considering the increasing new functionalities and devices in automotive electronics, it is crucial to efficiently utilize the available bandwidth.
- *Jitter:* This metric defines how much the transmitted periodic messages deviate from the periodicity.

The constraints and metrics described above were addressed in different articles. The basic guidelines to design the SM for a given message set are discussed in [9], and several scheduling strategies are proposed in [14]–[17] for TTCAN networks. The performance of the different proposed approaches is tested using two benchmark message sets, as well as self-constructed example message sets. The SAE benchmark message set [18] as introduced by the *Society of Automotive Engineers* (SAE) consists of 53 periodic and sporadic messages sent between seven different subsystems, such as batteries, brakes, and a vehicle controller in a prototype electric car. The PSA benchmark set is provided by the *Peugeot–Citroen Automobile Company* (PSA), and it was implemented in a prototype car. It consists of 12 periodic messages that are exchanged among six nodes [12].

A stochastic optimization algorithm for constructing the TTCAN SM is suggested in [14]. Their approach consists of a scheduling phase and an optimization phase to minimize the jitter. In the scheduling phase, a set of initial system matrices is constructed observing the average periods of messages in the schedule. These initial matrices are then optimized by applying a set of random transformations and keeping or discarding the resulting matrices according to the average jitter over the matrix duration. The approach is applied to the SAE and PSA benchmark sets (some of the results are obtained for modified versions of these message sets). The authors conclude that after a sufficient number of iterations (on the order of tens of thousands), matrices with small jitters can be found. No results or discussions are presented regarding the schedulability of the messages, bandwidth utilization, number of triggers, and sporadic messages.

A heuristic method for TTCAN scheduling is presented without a systematic approach in [16]. Rate monotonic scheduling is first applied for the periodic messages, and then, arbitrating windows are placed. The authors make sure that a certain trigger count per node is not exceeded. The approach is demonstrated on a self-constructed example message set. The authors conclude that using a long BC and evenly distributed arbitrating windows results in better performance for average message delay, jitter, and bus frequency response.

Similarly, a heuristic approach is proposed in [17]. SM is chosen as the least common multiple (LCM) of all the message periods. The schedulability of the messages in the arbitrating windows is checked by using the approach in [13] after placing the exclusive windows in the SM. The method is demonstrated on a set of messages used for an experimental mechatronic platform. The presented results focus on the schedulability of the SM.

A TTCAN scheduler called Smart-PLAN is presented in [15]. Messages are placed in the first BC according to their slacks (time left until deadline), and the column widths of the matrix are determined by this first BC allocation. The remaining messages are placed in the rest of the SM in the columns that they can fit according to their slack values. The approach is demonstrated using the SAE benchmark, and the results for bus utilization are presented. The jitter of periodic messages, sporadic messages, and the number of triggers per node are not considered in this paper.

C. Our Approach

In this paper, we give a formal discussion of the requirements and methods for scheduling messages on a TTCAN network. We first analyze the periodic messages and identify a condition on the period of these messages. We give a formal description, including performance metrics, of the scheduling problem and investigate optimal scheduling strategies (with zero jitter and a minimum network utilization and number of Triggers) for these periodic messages. Then, we examine the case where a set of periodic messages does not fulfill the stated condition. In Section IV, we present three systematic strategies to reduce the impact of such periodic messages on certain performance metrics. In particular, we propose *reduction of message periods*, *reduction of the matrix duration*, and *packing signals in frames*. Our method for periodic messages is complemented with an algorithm for scheduling sporadic messages in Section IV-C, and an example SM for the SAE benchmark message set is designed. Our approach is novel in both problem formulation and proposed solution. Different from other approaches, we analyze the properties of the message sets and the constraints to be fulfilled and construct feasible system matrices by considering multiple performance metrics rather than focusing on a single metric or constraint.

III. Investigation of Signal Properties

A. Definitions, Constraints, and Performance Metrics

We introduce the following notation to formally describe the problem under investigation.

We assume that a set of N *nodes* $\mathcal{N} = \{1, \ldots, N\}$ communicates on a TTCAN network, and we denote $\mathcal{S} = \{S_1, \ldots, S_F\}$ as the set of *signals* to be sent. For transmission on the network, signals are compiled to form a set of *messages* $\mathcal{M} = \{M_1, \ldots, M_G\}$, and we assume that more than one signal transmitted by the same node can be contained in a single message.[1] We introduce the maps $t_{\mathcal{M}} : \mathcal{M} \to \mathcal{N}$ and $r_{\mathcal{M}} : \mathcal{M} \to 2^{\mathcal{N}}$,[2] where $t_{\mathcal{M}}(M_m)$ is the *transmitter node*, and $r_{\mathcal{M}}(M_m)$ is the set of *receiver nodes* of the message $M_m \in \mathcal{M}$. Hence, we assume that broadcast or multicast of messages is possible. pm_m, dm_m, and $lm_m = 47 + 8\lceil bm_m/8\rceil + \lfloor(34 + 8\lceil bm_m/8\rceil/4)\rfloor$ denote the *period*, *deadline*, and *length* of the message M_m, respectively, where bm_m is the number of data bits of M_m. Messages can be periodic or sporadic, and we assume that for periodic messages $pm_m = dm_m$ and for sporadic messages $pm_m \geq dm_m$. We express message periods, deadlines, and lengths in multiples of the bit time τ_{bit}. Note that bit and τ_{bit} are used interchangeably fitting to the context.

We describe the SM according to Section II-A with its *BC width* B and its matrix *column widths* $\{C_1, \ldots, C_H\}$ expressed in τ_{bit}. Note that $\sum_{c=1}^{H} C_c = B$. We denote the *number of lines* of the SM by $L = 2^q$, $q \leq 6$. The *matrix cycle* $T := B \cdot L$ is the duration of the SM in τ_{bit}.

Each message $M_m \in \mathcal{M}$ has a set of Tx_Triggers $\mathcal{T}^m = \{Tx_1^m, \ldots, Tx_{K_m}^m\}$ and a set of Rx_Triggers $\mathcal{R}^m = \{Rx_1^m, \ldots, Rx_{L_m}^m\}$. The triggers indicate the starting time of the window in which a message is transmitted. Unless otherwise stated, the periodic messages are transmitted in exclusive windows, and sporadic messages are transmitted in the arbitrating windows. Each Tx_Trigger for M_m is located at the sending node $t_{\mathcal{M}}(M_m)$, and the corresponding Rx_Triggers are located at the receiving nodes $r_{\mathcal{M}}(M_m)$. For $Tx_t^m = (c_t^m, r_t^m, p_t^m)$, c_t^m is the TC of the Tx_Trigger, r_t^m is the line of its first occurrence (CO), and $2^{p_t^m}$ is the period of the message expressed in BCs (RF), where $0 \leq p_t^m \leq q$. Note that $Rx_t^m = Tx_t^m$ for periodic messages, and Rx_t^m is not defined for sporadic messages. Hence, the number of occurrences of M_m in the column that is indicated by a Tx_Trigger Tx_t^m is $2^{q-p_t^m}$. There can be more than one Tx_Trigger for any given message that differs by the TCs and possibly RFs, and there can be more than one message with the same Tx_Trigger for arbitrating windows.

Consider a message M_m with its related set of Tx_Triggers $\mathcal{T}^m$. Then the total amount of time allocated A_{M_m} and the total amount of time used for transmitting the signal data D_{M_m} for this message in one matrix cycle can be expressed as

$$A_{M_m} = \sum_{t=1}^{K_m} 2^{q-p_t^m} C_{c_t^m}$$
$$D_{M_m} = \frac{T}{pm_m} \cdot bm_m. \tag{2}$$

Our performance metrics are based on (2).

1) Utilization (U) and Matrix Load (ML): We provide two metrics that show the efficiency of bandwidth use instead of giving a single utilization metric. The *network utilization* U is the fraction of the allocated transmission time that is used for signal data transmission, i.e.,

$$U = \frac{\sum_{m=1}^{G} D_{M_m}}{\left(\sum_{m=1}^{G} A_{M_m}\right) + 95 \cdot 2^q} \tag{3}$$

where the term $95 \cdot 2^q$ represents the time for the transmission of the reference message in 2^q matrix lines. The *matrix load* ML is the fraction of the matrix cycle that is allocated for the transmission of messages

$$ML = \frac{\left(\sum_{m=1}^{G} A_{M_m}\right) + 95 \cdot 2^q}{T}. \tag{4}$$

[1]Formal methods to construct the set of messages from a given set of signals are discussed in Section IV-D.

[2]$2^{\mathcal{N}}$ is the power set of $\mathcal{N}$.

The network utilization indicates how efficiently the existing messages are scheduled in a given SM, whereas the matrix load represents the amount of new messages that can be added to an existing SM.

2) Jitter (J): The periodic messages are to be periodically delivered to the receiving nodes. Hence, ideally, the exclusive windows for a specific message should be placed in the SM such that the message is transmitted with the same period. Consider a periodic message M_m. Then, the time instants when M_m is scheduled with a Tx_Trigger $Tx_t^m \in \mathcal{T}_m$ are $(\sum_{k=1}^{c_t^m-1} C_K) + B \cdot r_t^m + j \cdot 2^{p_t^m} \cdot B, j = 0, \ldots, 2^{q-p_t^m}$.

Let $\{w_t^m, \ldots, w_{W_m}^m\}$ denote the ordered set of time instants in the matrix cycle that the message is scheduled with its Tx_Triggers, where $W_m := \sum_{t=1}^{K_m} 2^{q-p_t^m}$ is the number of exclusive windows allocated for the message M_m. Then, the *local jitter* J_K^m for each time instant w_K^m, $k = 1, \ldots, W_m$ is defined as the deviation of the intertransmission time at w_K^m from the actual message period pm_m, i.e.,

$$J_K^m = \begin{cases} |w_K^m - w_{k-1}^m - pm_m|, & \text{for } k \neq 1 \\ |w_1^m + T - w_{W_m}^m - pm_m|, & \text{otherwise} \end{cases}. \tag{5}$$

The average jitter J_m for M_m over the matrix cycle and the average jitter J for the entire SM are

$$J^m = \frac{\sum_{k=1}^{W_m} J_K^m}{T} \quad \text{and} \quad J = \sum_{m=1}^{G} J^m. \tag{6}$$

3) Trigger Count for Node n: We assume that the information for the reference message is stored in a separate trigger Ref_Trigger. Then, the number T_n of triggers for a node n is the sum of its Tx_Triggers, Rx_Triggers, and Ref_Trigger, i.e.,

$$T_n = \sum_{m, t_{\mathcal{M}}(M_m)=n} K_m + \sum_{m, n \in r_{\mathcal{M}}(M_m)} L_m + 1. \tag{7}$$

Note that, as discussed in Section II-B, there are practical hardware constraints that are both imposed by the TTCAN standard and depend on a specific TTCAN controller chip [9]. In particular, $L = 2^q$, $q \leq 6$, and $B \leq 2^{16}$, and the number of triggers per node (T_n) is smaller than 32, while the repetition period of a trigger has to be a power of 2. We also take the Tx_Enable period as defined in Section II-A2 to be 16 τ_{bit}.

B. Relationship Between Signal Periods and BC

To point out the issues related to the construction of the SM for a given set of nodes and an associated set of *periodic* messages, we first investigate under which conditions the matrix load and the trigger count are minimized with zero jitter while meeting all specified message deadlines.

1) Equal Message Length: We first assume that all messages have an equal length lm. Furthermore, we identify the smallest message period with pm_1 and require that the period of any message M_m can be represented as $pm_m = 2^j pm_1$ with $j \leq j_{\max} \leq 6$, i.e., all message periods are multiples of the smallest period, and the multiplier has to be a power of 2. We denote r_j as the number of messages with the respective period $2^j pm_1$.

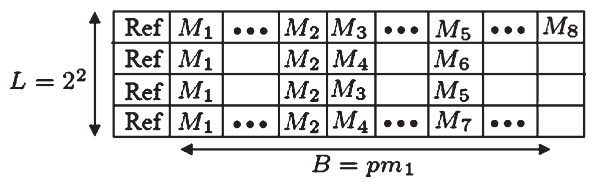

$L = 2^2$
Ref M_1 ••• M_2 M_3 ••• M_5 ••• M_8
Ref M_1 M_2 M_4 M_6
Ref M_1 M_2 M_3 M_5
Ref M_1 ••• M_2 M_4 ••• M_7 •••
$B = pm_1$

Fig. 2. SM construction.

Using the above assumptions, we propose the following SM construction: The BC length is $B = pm_1$; the number of lines is $L = 2^{j_{\max}}$; and we compute the sum as $M := \sum_{j=0}^{j_{\max}} r_j 2^{j_{\max}-j}$, which represents the number of messages transmitted in one matrix cycle. Including the Tx_Enable period of 16 τ_{bit} and the reference message with 95 τ_{bit}, the number of bits sent per matrix cycle is $M \cdot (lm + 16) + 95 \cdot 2^{j_{\max}}$. Formally, the following result can be stated.

Lemma 1: Assume that the message set described above is given. The set of messages is schedulable if and only if

$$(lm + 16) \cdot \sum_{j=0}^{j_{\max}} r_j 2^{j_{\max}-j} + 95 \cdot 2^{j_{\max}} \leq pm_1 2^{j_{\max}} = B \cdot L. \tag{8}$$

If the schedulability is given, the SM with the minimum number of G Tx_Triggers (i.e., one Tx_Trigger per message) is achieved by choosing $L = 2^{j_{\max}}$, $B = pm_1$, and the number of columns is $\lceil (M/2^{j_{\max}}) \rceil$.

Proof: We first show that the schedulability of the message set implies (8) by contradiction. Assume that

$$(lm + 16) \cdot \sum_{i=0}^{j_{\max}} r_j 2^{j_{\max}-j} + 95 \cdot 2^{j_{\max}} > pm_1 2^{j_{\max}}. \tag{9}$$

The number $pm_1 2^{j_{\max}}$ represents the largest message period, and thus every message with period $pm_1 2^j$ has to be transmitted $2^{j_{\max}-j}$ times during the time $pm_1 2^{j_{\max}}$. The sum of all messages to be transmitted during $pm_1 2^{j_{\max}}$ is thus $\sum_{i=0}^{j_{\max}} r_j 2^{j_{\max}-j}$ with message length $lm + 16$. Accounting for the $95 \cdot 2^{j_{\max}}$ reference message bits, the total number of bits transmitted in a matrix cycle is $(lm + 16) \cdot \sum_{i=0}^{j_{\max}} r_j 2^{j_{\max}-j} + 95 \cdot 2^{j_{\max}} > pm_1 2^{j_{\max}}$ because of the assumption in (9). Thus, the message set is not schedulable.

To show the reverse direction of the statement, we construct an SM that provides schedulability of the message set using the parameters $L = 2^{j_{\max}}$ and $B = pm_1$ and the column widths $C_c = lm + 16$ and $c = 1, \ldots, \lceil M/2^{j_{\max}} \rceil$. The Tx_Triggers for the columns are specified as follows.

- For each message with period $pm_1 \cdot 2^j$, $j \geq 1$, a Tx_Trigger with repetition period 2^j is used. 2^{q-j} such messages share a column.
- If the messages of a certain period do not complete a column, the column is filled with messages of a higher period, and the corresponding Tx_Triggers are introduced.

The construction of the SM is illustrated in Fig. 2. The messages M_1 and M_2 have the period pm_1; M_3, M_4, and M_5 have the period $2 \cdot pm_1$; and M_6, M_7, and M_8 have the period $2^2 \cdot pm_1$. The described construction procedure indeed results in $\lceil (M/2^{j_{\max}}) \rceil$ matrix columns with enough space for all messages and one Tx_Trigger per message. ■

Lemma 1 provides a schedulability result for a particular message configuration and proposes a systematic SM construction such that all message deadlines are met without jitter. The network utilization and the matrix load evaluate to

$$U = \frac{\sum_{m=1}^{G} 2^q \frac{pm_1}{pm_m} bm_m}{2^q \left(\sum_{m=1}^{G} \frac{pm_1}{pm_m} C_1 + 95 \right)}$$

$$ML = \frac{2^q \left(\sum_{m=1}^{G} \frac{pm_1}{pm_m} C_1 + 95 \right)}{T}.$$

This idea can be applied to the example constructed in [16], which exhibits the message properties required in this section. The smallest message period is 5 ms. As $\tau_{\text{bit}} = 2\ \mu$s, $pm_1 = 2500\ \tau_{\text{bit}}$. Furthermore, it is assumed that all messages have a fixed length of 8 B, which corresponds to a transmission time of 151 τ_{bit}. There are $r_0 = 3$, $r_1 = 5$, $r_2 = 4$, and $r_3 = 4$ messages with period pm_1, $2pm_1$, $2^2 pm_1$, and $2^3 pm_1$, respectively. Consequently, we choose $L = 2^3$ and $B = pm_1$, and $\lceil (M/2^{j_{\max}}) \rceil = \lceil (56/2^3) \rceil = 7$ columns are needed. The network utilization is $U = 3584/9216 = 39\%$, and the matrix load is $ML = (9216/2^3 \cdot 2500) = 46\%$. Note that Lemma 1 can also be applied to the message set in [17], i.e., an optimal schedule can be computed for the message sets in [16] and [17] using our method.

2) Different Message Lengths: The previous section only considers the case where each message has the same length. This assumption can be lifted by filling the matrix columns with different size messages, where the largest message determines the column width. It is readily observed that this strategy yields a more conservative schedulability result, as messages that are smaller than the respective column size use up bandwidth without transmitting data. In this section, we outline the construction of the SM with the optimal configuration, where the least amount of bits without data is transmitted.

Analogous to the considerations for an equal message length, the number of matrix columns for a given message set with r_j messages of period $2^j \cdot pm_1$, $s = 1, \ldots, j_{\max}$ evaluates to $\lfloor (\sum_{j=0}^{j_{\max}} r_j 2^{j_{\max}-j} / 2^{j_{\max}}) \rfloor$. Similar to the previous section, the number of Tx_Triggers needed to schedule all messages is equal to the number of messages $\sum_{j=0}^{j_{\max}} r_j$. Associating each of the required Tx_Triggers T_1^m to a message M_m, the repetition period of T_1^m is specified, i.e., $T_1^m = (c_1^m, r_1^m, pm_m/pm_1)$. Let $\mathcal{T}$ be the set of all Tx_Trigger combinations $\{T_1^1, \ldots, T_1^G\}$ that obey the specifications of the TTCAN standard for the given message set. We want to solve

$$\arg\max_{\tau \in \mathcal{T}} U = \max_{\tau \in \mathcal{T}} \frac{\sum_{m=1}^{G} D_{M_m}}{\sum_{m=1}^{G} A_{M_m} + 95 \cdot 2^q}. \tag{10}$$

TABLE II
MESSAGES FOR THE PSA EXAMPLE

messages M_i, $i =$	1	2	3	4	5	6,8,11	7	9	10	12
period ($pm_1\ \tau_{\text{bit}}$)	1	1	2	1	2	2^2	1	2	2^3	2^3
data (bit)	64	24	24	16	40	40	32	32	56	8

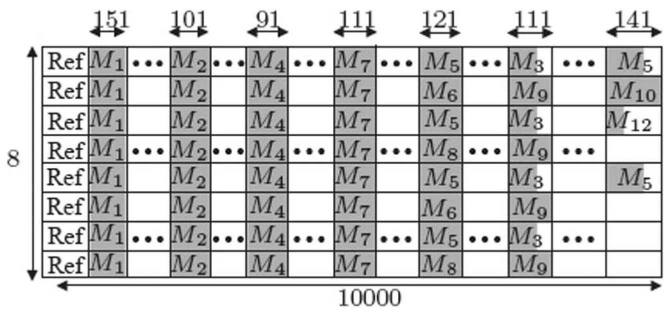


Fig. 3. SM for the PSA example.

The maximization can, for example, be evaluated by the exhaustive enumeration of all elements in $\mathcal{T}$ and determining the minimum value for the sum in (10) with the corresponding set of Tx_Triggers.

The procedure described above has been applied to the modified PSA example in [14] with the message set in Table II ($pm_1 = 10\,000\ \tau_{\text{bit}}$). An optimal configuration with a utilization of $U = (1264/5604) = 22.56\%$ and a matrix load of $ML = (5604/10\,000) = 7.01\%$ has been constructed, as shown in Fig. 3, where the utilization of the columns is indicated by the grey boxes. Note that although [14] suggests to provide optimized results, the network utilization (26.49%) and the matrix load (8.38%) differ from our results. Also, observe that the space between the allocated exclusive windows can be arbitrarily filled with arbitrating or free windows.

C. Limitations

The results stated in the previous section rely on the fact that all periods are multiples of the smallest period. In this section, the implication of lifting this assumption while keeping all the other requirements is discussed. That is, it is no longer true that all message periods can be represented as $2^j pm_1$ with a smallest period pm_1. To schedule the messages periodically without jitter, the matrix cycle T has to be an integer multiple of the LCM of all the message periods. It can thus be written as $T = 2^{j_{\max}-j} pm_m K$ for any message M_m, where j is chosen such that $2^{j_{\max}-j}$ contains all factors of 2 of a prime factorization of T/pm_m, and the *nonharmonic divisor* K does not contain any multiples of 2. Messages whose periods are either multiples or divisors of the BC length B can be scheduled in the same columns in each row where they appear. However, messages that do not fulfill the above property have to be scheduled in different columns to avoid jitter. For a detailed discussion, we distinguish messages M_m with $pm_m < B$ and $pm_m > B$.

1) $pm_m < B$: The *offset* o_m between the occurrences of M_m in consecutive lines of the SM is $o_m = \lceil (T/L \cdot pm_m) \rceil pm_m - (T/L)$ and can be written as $o_m =$

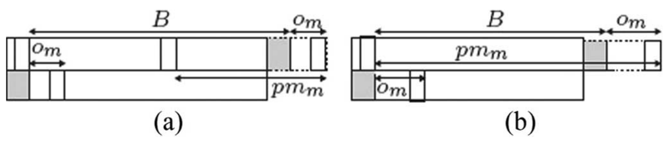


Fig. 4. Column offset.

TABLE III
PERIODIC MESSAGES FOR THE SAE EXAMPLE

message M_i $i =$	1,2,4,6	7	8,9	10	11	12	21
period (pm_1)	20	1	1	20	1	20	200
Tx/Rx	6/5	2/5	1/5	4/5	4/5	1/5	4/5
message M_i $i =$	3,5,13	32	29,30	33	36	42	43,49
period (pm_1)	200	1	2	200	200	1	1
Tx/ Rx	6/5	5/6	5/6	5/6	5/1	5/3	3/5

$\lceil(2^{j_{\max}-j}pm_mK/2^{j_{\max}} \cdot pm_m)\rceil\, pm_m - (2^{j_{\max}-j}pm_mK/2^{j_{\max}}) = (pm_m/2^j)(\lceil(K/2^j)\rceil 2^j - K)$ [see Fig. 4(a)]. Writing $K = 2^j \cdot a + b$, the offset is

$$o_m = \frac{pm_m}{2^j}\left((a+1)\cdot 2^j - a\cdot 2^j - b\right) = \frac{pm_m\cdot(2^j - b)}{2^j} \tag{11}$$

i.e., if $(2^j - b/2^j)$ is not an integer, then the message has to be transmitted in a different column in consecutive matrix lines if no jitter shall be introduced. In this case, the message can be scheduled in the same column in each 2^jth matrix line, and $(T/L)(2^j/pm_m) = (pm_m \cdot K \cdot 2^{j_{\max}-j} \cdot 2^j / 2^{j_{\max}} \cdot pm_m) = K$ Tx_Triggers are needed.

2) $pm_m > B$: The *offset* o_m between the occurrences of the message in different lines of the SM is $o_m = pm_m - \lfloor(pm_m \cdot L)/T\rfloor T/L$ and can be written as $pm_m(1 - \lfloor(pm_m \cdot 2^{j_{\max}} / pm_m \cdot K \cdot 2^{j_{\max}-j})\rfloor (pm_m \cdot K \cdot 2^{j_{\max}-j} / pm_m \cdot 2^{j_{\max}})) = (pm_m/2^j)\ (2^j - \lfloor(2^j / K)\rfloor K)$ [see Fig. 4(b)]. Writing $2^j = K \cdot c + d$, the resulting offset is

$$o_m = \frac{pm_m}{2^j}(K\cdot c + d - c\cdot K) = \frac{pm_m\cdot d}{2^j}. \tag{12}$$

If $pm_m \cdot d/2^j$ is not an integer, then M_m has to be sent in different columns and can be scheduled in the same column in each 2^jth matrix line with $(T/L)(2^j/pm_m) = K$ Tx_Triggers.

The following SAE benchmark example introduced in [18] manifests the issues pointed out in this section.

D. Application to Benchmark Examples

The parameters relevant for the discussion of the SAE example are summarized in Table III. (Note that the original message identifications are used.) All messages carry at most 1 B of data, and the smallest message period is $pm_1 = 2500\ \tau_{\text{bit}}$ at a network speed of 500 kb/s.

Following the above discussion, we choose the LCM of the periodic messages as the matrix cycle $T = 500\,000\ \tau_{\text{bit}}$. As $(T/pm_1) = (500\,000/2500) = 200 > 2^6 = 64$, it is the case that $pm_1 < (T/2^{j_{\max}})$. It holds that $T = pm_1 \cdot 2^3 \cdot 25$, i.e., $2^{j_{\max}-j} = 2^3$ with the nonharmonic divisor $K = 25$. The choice of $L = 2^{j_{\max}} = 2^5$ results in $2^j = 2^2$, and thus, $K = 25 = 6 \cdot 2^2 + 1$. According to Section III-C1, $(2^j - b/2^j) = (3/4)$, and the message M_1 has to be scheduled in different columns in four consecutive lines. Furthermore, $K = 25$ Tx_Triggers are needed for each message with period pm_1. Similarly, for messages with period $pm_m = 20 \cdot pm_1$, it holds that $20 \cdot pm_1 > B$, according to Section III-C2. We compute $T = 20 \cdot pm_1 \cdot 2 \cdot 5$, i.e., $2^{j_{\max}-j} = 2$ and $K = 5$. Then, $L = 2^5$ implies that $2^j = 2^4 = 3 \cdot K + 1$. The offset is $(20 \cdot pm_1 \cdot 1/2)^4 = (5/4) \cdot pm_1$, and $K = 5$ Tx_Triggers are needed to schedule these messages. Evaluating (4), the matrix load for periodic messages of the SAE example is 24.3%.

Looking at node 1, two messages with period pm_1 and one message with period $20pm_1$ have to be transmitted, i.e., already, $2 \cdot 25 + 5 = 55$ Tx_Triggers are needed. It can be concluded that the requirement of zero jitter leads to very high Tx_Trigger counts for messages that exhibit a nonharmonic divisor K different from 1.

In the next section, we address the above issues and develop different systematic methods to construct the SM if the properties required in Section III-B2 are not fulfilled. In particular, we consider the following ideas.

- Message periods can be reduced to fulfill the requirements listed in Section III-B2. This increases the matrix load and introduces jitter.
- The number of occurrences of messages with the smallest period (and thus the number of Tx_Triggers) can be reduced by the choice of a smaller matrix cycle. This increases the matrix load.
- The number of messages can be reduced by packing signals with equal periods that are transmitted by the same node into one message. This decreases the network load but possibly increases the number of Rx_Triggers.

IV. SYSTEMATIC SM CONSTRUCTION

A. Reduced Message Periods

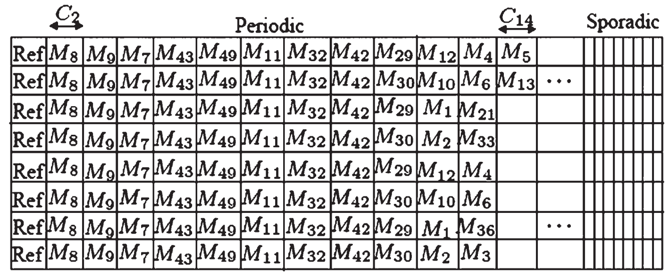
The number of Tx_Triggers in the above example is large as the requirement of zero jitter leads to exclusive windows in different matrix columns for one message (see Fig. 4). If the trigger count has to be reduced, one idea is to realign exclusive windows in consecutive matrix lines while lifting the zero jitter requirement. Considering a message M_m with $pm_m \le B$ and the offset $o_m = pm_m(2^j - b/2^j)$, according to (11), the jitter in lines $k = 2, \ldots, 2^j$ is $(B/pm_m) \cdot k \cdot o_m$, and thus, the overall jitter per matrix cycle $(T = 2^j \cdot 2^{j_{\max}-j} \cdot B)$ is

$$J^m = \frac{B\cdot pm_m\cdot(2^j - b)}{2^{j_{\max}}\cdot B\cdot pm_m\cdot 2^j}\sum_{k=1}^{2^j-1} k = \frac{(2^j - b)(2^j - 1)}{2^{j_{\max}+1}}. \tag{13}$$

Similarly, the jitter for a message with $pm_m > B$ and offset $o_m = pm_m \cdot d/2^j$ with $B \cdot 2^j/pm_m$ occurrences in 2^j consecutive lines is

$$J^m = \frac{d}{2^{j_{\max}+1}}(K+1). \tag{14}$$

Note that the Tx_Trigger count is reduced from K to $\lceil(B/pm_m)\rceil$ for $pm_m \le B$ and to 1 for $pm_m > B$. The idea of adjusting message periods to the BC length can indeed be used to reduce the trigger count if the resulting jitter is tolerable.



	C_2					Periodic						C_{14}		Sporadic
Ref	M_8	M_9	M_7	M_{43}	M_{49}	M_{11}	M_{32}	M_{42}	M_{29}	M_{12}	M_4	M_5		
Ref	M_8	M_9	M_7	M_{43}	M_{49}	M_{11}	M_{32}	M_{42}	M_{30}	M_{10}	M_6	M_{13}	$\cdots$	
Ref	M_8	M_9	M_7	M_{43}	M_{49}	M_{11}	M_{32}	M_{42}	M_{29}	M_1	M_{21}			
Ref	M_8	M_9	M_7	M_{43}	M_{49}	M_{11}	M_{32}	M_{42}	M_{30}	M_2	M_{33}			
Ref	M_8	M_9	M_7	M_{43}	M_{49}	M_{11}	M_{32}	M_{42}	M_{29}	M_{12}	M_4			
Ref	M_8	M_9	M_7	M_{43}	M_{49}	M_{11}	M_{32}	M_{42}	M_{30}	M_{10}	M_6			
Ref	M_8	M_9	M_7	M_{43}	M_{49}	M_{11}	M_{32}	M_{42}	M_{29}	M_1	M_{36}		$\cdots$	
Ref	M_8	M_9	M_7	M_{43}	M_{49}	M_{11}	M_{32}	M_{42}	M_{30}	M_2	M_3			

Fig. 5. SM for the SAE benchmark message set.

Moreover, the above results can be viewed from a different perspective. The jitter J^m depends on the nonharmonic divisor K either directly in (14) or indirectly via b or d in (13) and (14). Hence, to avoid jitter, it is desirable to have a BC length B that is either a multiple or a factor of the different message periods. This observation is further exploited in the next section.

B. Reduction of the Matrix Cycle

To keep the number of Triggers small, it is desired that the nonharmonic divisor $K = (T/2^{j_{\max}-j} pm_m)$ is small, and thus, j is as small as possible.

The benchmark example in Section III-D requires a matrix cycle T that is 200 times larger than the smallest message period pm_1, which results in a comparably long BC and a large number of Tx_Triggers. This effect can be circumvented by choosing the matrix cycle T smaller than the LCM of the message periods, such that the messages M_m with smaller periods than T, i.e., $pm_m \leq T$, can be represented as $T/2^j$. In that case, we define $L = (T/pm_1) = 2^{j_{\max}}$ and schedule the messages with smaller periods than T according to the strategy in Section III-B. To address the messages M_m with periods larger than T, it can be observed that such messages have to be scheduled with the period of the *greatest common divisor* (gcd), $\gcd(T, pm_m)$, of the matrix cycle and the message period to be transmitted at the correct time instants (i.e., without jitter).

This idea is applied to the SAE example in Section III-D. We choose $T = 20\,000\ \tau_{\text{bit}} = 2^3 \cdot pm_1 = 2^2 \cdot 2pm_1 = 2^1 \cdot 4pm_1$ as the matrix cycle. The number of lines is thus $L = 2^3$. Noting that all messages have the same length, the messages with periods pm_1, $2pm_1$, and $4pm_1$ can be scheduled as in Section III-B1. The result can be seen in the first ten columns of the SM in Fig. 5. Messages with periods $200pm_1$ and $20pm_1$ have to be scheduled with the periods $\gcd(T, 200pm_1) = T$ and $\gcd(T, 20pm_1) = T/2$, respectively. The required 13 columns for the periodic messages of the SAE example are depicted in Fig. 5 (the columns are placed next to each other for illustration purposes). Note that only one Tx_Trigger and one Rx_Trigger are required for each message. The node with the largest amount of triggers is node 6 (vehicle controller) with seven Tx_Triggers and 13 Rx_Triggers.

The above discussion suggests that the more frequent transmission of messages M_m with a period that is larger than the matrix cycle increases the matrix load. For example, the messages with period $200pm_1$ are now scheduled in every matrix cycle, but a new value is only transmitted in every 25th matrix cycle, which implies that 24 of the 25 matrix cycles are unused. Assuming that the Tx_Trigger for M_m is $Tx_1^m = (c_1^m, r_1^m, p_1^m)$, the additional load for such messages can be computed as $(T/\gcd(T, pm_m) - T/pm_m)C_{c_1^m}$, i.e., the difference of the number of scheduled exclusive windows and the number of used exclusive windows per matrix cycle multiplied with the respective column length. This formula evaluates to $(24/25)C_{c_t^m}$ for messages with period $200pm_1$ and $(8/5)C_{c_t^m}$ for messages with period $20pm_1$. Although more unused data are transmitted over the network, the number of Tx_Triggers and Rx_Triggers per node is decreased to a very large extent compared to the case with matrix cycle $T = 500\,000\ \tau_{\text{bit}}$, and all periodic messages can be scheduled without jitter. Fig. 5 depicts the periodic part of the resulting SM. The network utilization evaluates to $U = 9.47\%$, and the matrix load is 31.46%.

C. Scheduling of Sporadic Messages

After proposing different scheduling methods for periodic messages, we now consider the scheduling problem for sporadic messages that are given as a set $\mathcal{M} = \{M_{1,1}, \ldots, M_{1,r_1}, M_{2,1}, \ldots, M_{2,r_2}, \ldots, M_{g,r_g}\}$, where r_j is the number of messages with the deadline dm_j, and g denotes the number of different deadlines. Different from periodic messages, sporadic messages can have deadlines that are smaller than their period. Note that if dm_j is larger than the matrix cycle T, then we reduce it to T.

The TTCAN network must be able to transmit a sporadic message with a deadline dm_j within dm_j after it is generated. As there is no information about the time instant when the message can be generated, this can be achieved by providing an arbitrating window for this message every *transmission period* dm_j. Since the number of Tx_Triggers is limited, we follow a column-oriented approach and choose the transmission periods that meet the deadlines of the messages according to the BC length B. We introduce the *line count* I_j that represents the largest number of BCs that fit into one transmission period. We also introduce the *column count* O_j that indicates the number of matrix columns needed to schedule one message with period dm_j in I_j matrix lines, i.e.,

$$I_j = \begin{cases} \lfloor \frac{dm_j}{B} \rfloor, & \text{if } dm_j \geq B \\ 1, & \text{otherwise} \end{cases}$$
$$O_j = \begin{cases} 1, & \text{if } dm_j \geq B \\ \lceil \frac{B}{dm_j} \rceil, & \text{otherwise} \end{cases} . \tag{15}$$

Note that we allocate the bandwidth for the sporadic messages to be transmitted with period dm_j. However, the messages do not arrive with the same period. Hence, the unused transmission instants can be utilized to accommodate messages with larger deadlines. This idea can be illustrated as follows. Consider a message $M_{1,1}$ with the smallest deadline dm_1 and line count I_1, i.e., the column count is O_1. The total number of arbitrating windows to be reserved for these messages per I_1 matrix lines is $I_1 \cdot O_1$. However, maximally, $\lceil (I_1 \cdot B/pm_{1,1}) \rceil$

occurrences of such messages with period $pm_{1,1}$ are possible during the corresponding time duration. Consequently

$$I_1 \cdot O_1 - \left\lceil \frac{I_1 \cdot B}{pm_{1,1}} \right\rceil \tag{16}$$

windows are free for messages with larger deadlines, assuming that deadline monotonic scheduling is applied among the messages that are assigned to the same arbitrating windows. Although messages with a smaller deadline always have priority over messages with a larger deadline, their period limits their maximum number of occurrences in a given interval of time. This guarantees the timely transmission of messages with a larger deadline in the remaining windows. Note that deadline monotonic scheduling can be realized by accordingly assigning the frame identifiers.

Based on the above considerations, we propose the following scheduling algorithm for sporadic messages:

```
Scheduling Algorithm for Sporadic Messages:
1) set $i := 1; c := 0; y_1 := 0; \ldots ; y_g := 0; x := 1$
2) if $x > g$
   $c_{\text{final}} = c$
   terminate
   if $i > g$
   $y_1 := y_1 + \kappa_{c,1}; \cdots ; y_g := y_g + \kappa_{c,g}$
   $x := 1; i := 1;$
3) compute the maximal $z \leq r_i$ such that
   $\lceil (I_i \cdot B/pm_{x,y_x+1}) \rceil + \cdots + \lceil (I_i \cdot B/pm_{x,y_x+\kappa_{c,x}}) \rceil +$
   $\cdots + \lceil (I_i \cdot B/pm_{i,y_i+z}) \rceil \leq O_x I_i$.
   If $z = 0$ and $x = i$
      $x = x + 1$
   else if $z \neq 0$ and $x = i$
      $c = c + O_x; \kappa_{c,i} = z$
   else
      $\kappa_{c,i} = z$
   for each $j = 1, \ldots, \kappa_{c,i}$
      $O_x$ Tx_Triggers for $M_{i,y_i+j}$, with $Tx_K^{i,y_i+j} = (c - k, 1, 0)$ and $k = 0, \ldots, O_x - 1$
   $i := i + 1$
   go to 2).
```

The variables i, c, x, and y_j keep track of the current deadline, the last column introduced in the matrix, the deadline that determines the current column count, and the index of the last message that has been scheduled for each deadline dm_j in the previous iteration, respectively. The algorithm is initialized with the smallest deadlines ($i = x = 1$) and with no columns ($c = 0$, and $y_1 = \cdots = y_g = 0$). In each iteration (starting from step 2), it is checked if the current column is already filled ($i > g$). If this is the case, which messages have already been scheduled ($y_1 := y_1 + \kappa_{c,1}; \ldots ; y_g := y_G + \kappa_{c,g}$) are stored, and the generation of new columns is prepared ($x := i := 1$). Otherwise, it is checked how many new messages ($\kappa_{c,i}$) with deadlines dm_i can be accommodated in the arbitrating windows generated for a message with deadline dm_x. $z = 0$ and $x = i$ represent the case where no new column can be added yet as no new message to be scheduled has been found. A new column is added if $z \neq 0$ and $x = i$. For each message M_{i,y_i+j} that is added to the O_x columns under consideration, a Tx_Trigger with RF 1 is added in each of the columns. The algorithm terminates if all messages have been scheduled ($x > g$). Together, scheduling the sporadic messages according to the above algorithm requires c_{final} columns in the SM. The sporadic message set is schedulable if (4) for the resulting set of Tx_Triggers evaluates to a matrix load of less than 100%, where the respective column width is determined by the largest message in the column.

We now apply this approach to the SAE set [18] with 31 sporadic signals with $dm_1 = 2500\ \tau_{\text{bit}}$, $r_1 = 1$, $pm_{1,1} = 25\,000\ \tau_{\text{bit}}$, $dm_2 = 10\,000\ \tau_{\text{bit}}$, $r_2 = 30$, $pm_{2,1} = 10\,000\ \tau_{\text{bit}}$, and $pm_{2,k} = 25\,000\tau_{\text{bit}}$ for $k = 2, \ldots, 31$. There are two different deadlines; hence, $g = 2$ holds. As we want to construct the SM for the complete SAE set, we adopt the values $B = 2500\ \tau_{\text{bit}}$ and $L = 2^3$ from Section IV-B.

In step 1 of the algorithm, $I_1 = \lfloor (2500/2500) \rfloor = 1$, and $O_1 = \lceil (1/1) \rceil = 1$. The $z = 1$ message with deadline dm_1 can be fitted in the first column $c = 1$ as $\lceil (4 \cdot 2500/25\,000) \rceil = 1$. Thus, $\kappa_{1,1} = 1$. The messages with the next larger deadline are considered in step 2 of the algorithm. These messages require $I_2 = \lfloor (10\,000/2500) \rfloor = 4$ rows. We now check how many of these messages can be fitted in the same column with the message that was placed in the first step. We find $(\lceil (4 \cdot 2500/25\,000) \rceil) \cdot 1 + \lceil (4 \cdot 2500/10\,000) \rceil) \cdot 1 + \lceil (4 \cdot 2500/25\,000) \rceil)2 \leq 4$, which means that three such messages can be placed in the same column as the first message. One Tx_Trigger with RF 1 is placed in the first column for each of the messages considered up to now. The remaining 27 messages with deadline dm_2 can be placed in seven new matrix columns in the last iterations of the algorithm with their respective Tx_Triggers. In total, eight matrix columns are reserved for the sporadic messages. The *best-case* network utilization according to (3) (assuming message arrival with the respective period) is $U = 1\%$, and the matrix load is $ML = 25.92\%$. The SAE message set in [18] suggests that node 6 has the largest amount of 11 Tx_Triggers for sporadic messages. This result together with the matrix load for periodic messages in Section III-D ($ML = 24.3\% + 25.9\% = 50.2\%$) can be compared to the matrix load derived in [15] (50.8%). Although no jitter is introduced by our systematic approach, a smaller matrix load is achieved.

Furthermore, the trigger count can be reduced by combining the results of this section with Section IV-B. The overall SM for the SAE example with BC length $B = 2500\ \tau_{\text{bit}}$ and number of lines $L = 8$ is shown in Fig. 5. All messages can be scheduled with a matrix load of 57.38%. Note that the columns for the periodic and sporadic messages are separated for illustration purposes. Generally, they can be placed arbitrarily. The node with the largest number of triggers is node 6 with $31 + 1 = 32$ triggers.

D. Packing Signals in Frames

The framing overhead for CAN messages, as indicated in (1), can be decreased by packing multiple signals in the same frame instead of individual frames [19], [20]. The application of such approaches for TTCAN is very beneficial, as the resulting smaller message set also requires a smaller number of triggers. In this section, we discuss methods to efficiently pack signals into frames.

We consider the set of signals $\mathcal{S}$, as introduced in Section III-A, and we call ps_m as the *period*, ds_s as the *deadline*, and ls_s as the *length* of a signal $S_s \in \mathcal{S}$ expressed as multiples of the bit time τ_{bit}. To associate signals to nodes, we define the two maps $t_\mathcal{S} : \mathcal{S} \to \mathcal{N}$ and $r_\mathcal{S} : \mathcal{S} \to 2^\mathcal{N}$ for the set of signals, where $t_\mathcal{S}(S_s)$ denotes the *sender node*, and $r_\mathcal{S}(S_s)$ denotes the set of *receiver nodes* of the signal $S_s \in \mathcal{S}$.

We define the relationship between signals and messages by the map pack: $\mathcal{M} \to 2^\mathcal{S}$ such that the pack(M_m) represents the set of signals packed into the message $M_m \in \mathcal{M}$. It is required that all signals in a message have the same transmitter node. Based on signal maps $t_\mathcal{S}$ and $r_\mathcal{S}$, we define the maps $t_\mathcal{M} : \mathcal{M} \to \mathcal{N}$ and $r_\mathcal{M} : \mathcal{M} \to 2^\mathcal{N}$ introduced in Section III-A. The transmitter node for a message $M_m \in \mathcal{M}$ is $t_\mathcal{S}(S_s)$ for any $S_s \in \text{pack}(M_m)$, and the set of receiver nodes of M_m is $r_\mathcal{M}(M_m) := \bigcup_{S_s \in \text{pack}(M_m)} r_\mathcal{S}(S_s)$. We denote $pm_m = \min_{S_s \in \text{pack}(M_m)} p_s$ as the *period*, $dm_m = \min_{S_s \in \text{pack}(M_m)} d_s$ as the *deadline*, and $lm_m = 47 + 8\lceil bm_m/8 \lceil \lfloor (34 + 8\lceil bm_m/8\rceil/4) \rfloor$ as the *length* of the message M_m, where $bm_m = \sum_{S_s \in \text{pack}(M_m)} l_s$ is the number of data bytes. Note that $lm_m < \sum_{S_s \in \text{pack}(M_m)}(47 + 8\lceil l_s/8\rceil + \lfloor(34 + 8\lceil l_s/8\rceil/4)\rfloor)$.

We first investigate a simple packing policy for periodic signals, where only the signals with the same periods and deadlines are packed together in a frame. The advantage of packing signals in messages can be illustrated by means of the number of receiver nodes $|r_\mathcal{M}(M_m)|$ of a message M_m. If all of the signals in this message are transmitted in individual frames, the transmitting node requires $|\text{pack}(M_m)|$ Tx_Triggers, and each receiver node requires at least one Rx_Trigger. If the signals are packed in one message M_m, the transmitter node requires only one Tx_Trigger, and each receiver node requires only one Rx_Trigger.

The signals with larger deadlines can be further packed into a message with a smaller deadline after applying simple packing. This second packing only decreases the overhead and Tx_Trigger count; however, it does not decrease the size of the message set or the number of Rx_Triggers any more. In addition to that, as the minimum signal deadline determines the deadline of the message, this modified packing increases the number of messages with smaller deadlines.

Observing that a node n with a periodic message M_m whose period is pm_m has an exclusive window to transmit at most every pm_m, it is possible to piggyback sporadic signals that have deadlines larger than pm_m and that are also transmitted by node n. This third packing idea further decreases the framing overhead by using less number of frames for the entire signal set, as well as decreasing the Tx_Trigger and Rx_Trigger counts. In addition, it decreases the number of sporadic signals that are transmitted in arbitrating windows. The disadvantage of this approach is that the time for the transfer of the sporadic signal bits is allocated with the periodic signal and, thus, is wasted if the sporadic signal is not transmitted frequently enough. Hence, the possible increase in utilization by piggybacking sporadic signals depends on the average period of the sporadic signal, as well as the relation of the periods of the periodic signal and the sporadic signal that is piggybacked on this signal. More precisely, the closer the periods are, the less bandwidth is wasted.

The simple packing idea has been applied to the periodic messages of the SAE message set. After packing, five messages with period pm_1, one message with period $2pm_1$, three messages with period $20pm_1$, and three messages with period $200pm_1$ are scheduled. Applying the method in Section IV-B with $B = pm_1$ and $L = 2^3$, the messages can be scheduled in seven columns. The network utilization is $U = 10.78\%$, and the matrix load is $ML = 27.62\%$. The highest trigger count for periodic messages of node 6 is reduced to 12.

Consequently, combining the systematic methods for the SM construction developed in this paper, the SAE benchmark message set can be scheduled with an overall matrix load (periodic and sporadic messages) of 53.54% and a maximal number of $12 + 11 + 1 = 24$ triggers for node 6.

Together, it can be concluded that the adjustment of the matrix cycle in Section IV-B and the simple packing idea in this section constitute the most favorable scheduling strategies for periodic messages if the periods cannot be represented as a multiple of the smallest period in the message set and a power of 2. Employing the presented methods, a slight increase in the network utilization allows for an effective reduction of the trigger count without introducing jitter.

V. Conclusion

TTCAN has been developed as a time-triggered in-vehicle network that is built on the existing CAN network standard and is fully compatible with CAN nodes. It provides a feasible transition to time-triggered operation that is required for predictable and fast network response to applications such as x-by-wire. The message schedule for TTCAN is constructed offline and has to guarantee the real-time schedulability of the messages as well as compliance with the controller hardware. Network performance metrics such as bandwidth utilization, network load, and message jitter are also determined by this message schedule. In this paper, we have presented a formal framework for the TTCAN message schedule construction, including the aforementioned performance metrics. This framework allows for a formal analysis of the message properties and constraints for constructing the TTCAN schedule for a given message set. Our analysis provides guidelines for designing message sets with desirable properties such as periods that are multiples of a smallest message period and powers of 2 that yield efficient schedules. In addition, we discuss what changes if the message set does not exhibit the desired properties. Based on the derived results, we propose a systematic approach to construct feasible schedules that guarantee the deadlines of the messages, respect the hardware constraints, utilize the bandwidth efficiently, and introduce as small jitter as possible. Our approach covers both periodic and sporadic messages, and we provide a feasible message schedule for the complete SAE benchmark message set.

Although we consider specific constraints that arise from the TTCAN operation, the general formulation of our methodology is promising to be applicable to other time-triggered networks. Our future work includes extending our approach to other time-triggered networks such as FlexRay. Furthermore, a software tool that automatically constructs the message schedule according to our scheduling strategies will be developed, and simulation studies, including best-effort messages without real-time requirements, will be carried out to investigate the average performance of the network.

References

[1] A. Albert, "Comparison of event-triggered and time-triggered concepts with regard to distributed control systems," in *Proc. Embedded World*, 2004, pp. 235–252.

[2] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in automotive communication systems," *Proc. IEEE*, vol. 93, no. 6, pp. 1204–1223, Jun. 2005.

[3] *LIN Specification Package*, Sep. 2003. revision 2.0. [Online]. Available: http://www.lin-subbus.org/

[4] *Time-Triggered Protocol TTP/C, High-Level Specification Document*, Nov. 2003. protocol version 1.1. [Online]. Available: http://www.tttech.com

[5] J. Berwanger, M. Peller, and R. Griegbach, *A New High Performance Data Bus System for Safety Related Application*, 1999. [Online]. Available: http://www.byteflight.com/specification

[6] *FlexRay Communication System*, Jun. 2004. protocol specification, version 2. [Online]. Available: http://www.flexray.com

[7] *CAN in Automation*, May 2005. [Online]. Available: http://www.cancia.org/can/

[8] G. Cena, A. Valenzano, and S. Vitturi, "Advances in automotive digital communications," *Comput. Standards Interfaces*, vol. 27, no. 6, pp. 665–678, Jun. 2005.

[9] G. Leen and D. Heffernan, "TTCAN: A new time-triggered controller area network," *Microprocess. Microsyst.*, vol. 26, no. 2, pp. 77–94, Mar. 2002.

[10] *TTCA*, May 2005. [Online]. Available: http://www.cancia.org/can/ttcan/

[11] *Road Vehicles Controller Area Network (CAN) Part 4: Time-Triggered Communication*, Std. ISO IS 11 898-4, 2004.

[12] N. Navet, Y.-Q. Song, and F. Simonot, "Worst-case deadline failure probability in real-time applications distributed over controller area network," *J. Syst. Archit.*, vol. 46, no. 7, pp. 607–617, Apr. 2000.

[13] K. Tindell, A. Burns, and A. J. Wellings, "Calculating controller area network (CAN) message response times," *Control Eng. Pract.*, vol. 3, no. 8, pp. 1163–1169, Aug. 2000.

[14] J. Fonseca, F. Coutinho, and J. Barreiros, "Scheduling for a TTCAN network with a stochastic optimization algorithm," in *Proc. Int. CAN Autom. Conf.*, 2002.

[15] M. Naughton and D. Heffernan, "SMART-plan: A new message scheduler for real-time control networks," in *Proc. IEE Irish Signals Syst. Conf.*, 2005, pp. 302–307.

[16] A. Albert and R. Hugel, "Heuristic scheduling concepts for TTCAN networks," in *Proc. Int. CAN Autom. Conf.*, 2005.

[17] R. Johannson, "Time and event triggered communication scheduling for automotive applications," Chalmers Lindholmen Univ. College, Goteborg, Sweden, Tech. Rep. 17, 2004.

[18] SAE paper J2056/1 June 93, Class C Application Requirements, SAE Handbook, vol. 2, pp. 23.366–23.272, Soc. Automotive Engineers, Warrendale, PA, 1994.

[19] C. Norstrom, K. Sandstrom, and M. Ahlmark, "Frame packing in real-time communication," in *Proc. 7th Int. Conf. RTCSA*, 2000, pp. 399–403.

[20] R. Saket and N. Navet, "Frame packing algorithms for automotive applications," *J. Embed. Comput.*, vol. 2, no. 1, pp. 93–102, 2006.

Klaus Schmidt received the M.S. and Ph.D. (mit Auszeichnung, with distinction) degrees from the University of Erlangen-Nuremberg, Erlangen, Germany, in 2002 and 2005, respectively, both in electrical, electronics, and communication engineering.

He is currently a Post-Doctoral Researcher with the Institute of Control and Automation, University of Erlangen. His research interests include controller synthesis for discrete event systems, networked control systems, and vehicular communication networks.

Ece G. Schmidt received the B.S. degree in electrical and electronics engineering from the Middle East Technical University, Ankara, Turkey, in 1997 and the M.S. and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 2001 and 2004, respectively.

She is currently a Faculty Member with the Department of Electrical and Electronics Engineering, Middle East Technical University. Her research interests include high-speed networks, networked control systems, and vehicular communication networks.