Optimal TDMA Time Slot and Cycle Length Allocation for Hard Real-Time Systems

Ernesto Wandeler Lothar Thiele
Computer Engineering and Networks Laboratory
Swiss Federal Institute of Technology (ETH) Zurich, Switzerland
E-mail: {wandeler,thiele}@tik.ee.ethz.ch

Abstract— We present an analytic method to determine the provably smallest possible slot length that must be allocated in a TDMA resource, to serve an event-triggered hard real-time load with arbitrary deterministic timing behavior. Based on this method, we then present constructive methods to find all feasible as well as the optimal cycle length in a TDMA resource, and we show how to determine the minimum required bandwidth of a TDMA resource. We demonstrate the applicability and computational efficiency of the presented methods in a case study of a large distributed embedded system with a TDMA bus, where we will find the optimal parameter set for the TDMA bus.

I. Introduction

In large distributed embedded systems, TDMA scheduling policies play an increasingly important role. TDMA is often employed in such systems on backbone communication resources that typically interconnect a large number of the present embedded computing units (ECU's), as shown in Fig. 1. This trend can best be observed in the area of safety-critical automotive and avionic systems, where TDMA-based communication protocols such as TTP, or more recently the mixed TDMA/FTDMA-based FlexRay replace more and more the formerly omnipresent CAN protocol. But also for communication on MpSoC's [7], as well as to provide QoS guarantees in network on chips [6], TDMA gets increasingly important.

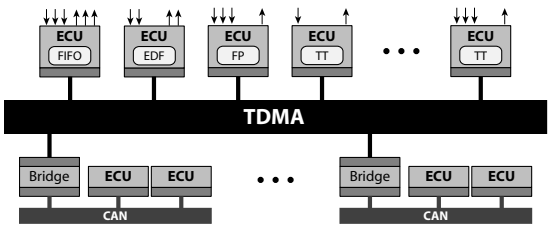

Fig. 1. A distributed system with TDMA communication.

TDMA protocols possess a number of advantages compared to event-triggered communication protocols such as CAN, that make them interesting for the use in such systems. First of all, TDMA resources support temporal composability, by clearly separating resource access of different subsystems that therefore do not interfere with each other. Moreover, TDMA resources have a very deterministic timing behavior, can be made fault tolerant, and support error detection, as well as error contention, i.e. a faulty subsystem does not affect the correct behavior of the remaining system. Note, that due to these properties, TDMA is also often applied for single processor scheduling, for example to enable composable and hierarchical scheduling, see e.g. [13].

A major difficulty that arises however during the design process of systems with TDMA-scheduled resources is parameter selection for the TDMA resources. Customizable parameters are typically the total bandwidth B of the resource, the cycle length c of the TDMA round, as well as the individual slot lengths s_i for the different service consumers of the TDMA resource.

In purely time-triggered systems, an optimal communication schedule that defines slot and cycle lengths can be constructed at design-time [8], but in reality heuristics are often used to find a valid communication schedule due to the computational complexity of finding the optimal schedule.

Many large distributed embedded systems are however not anymore designed as purely time-triggered systems, but contain instead mixed time- and event-triggered components. Be it because of the coexistence of time- and event-triggered subsystems (clusters) that are connected with each other by bridges, as considered in [12], or be it because of the existence of some event-triggered ECU's, as considered for example in [10] . Both is shown in Fig. 1.

When we get to such mixed time- and event-triggered systems, parameter selection for TDMA resources gets even more challenging. In [11], slot lengths are chosen as a fraction of a fixed cycle length, such that every service consumer with event-triggered load receives an individual total bandwidth from the TDMA resource. While the method presented in [11] can be used for systems with non-real-time event-triggered loads, it is only applicable by trial-and-error for systems with real-time event-triggered loads, i.e. loads with deadline constraints. For such systems, [12] presents a heuristic to assign slot lengths of a TDMA resource with fixed bandwidth and cycle length. [12] however only deals with strictly periodic loads and does therefore not consider buffering effects that may occur when serving loads with jitter or bursts.

Very recently, a method for slot as well as cycle length optimization based on evolutionary search techniques was presented in [7]. This method can be used to parameterize slot and cycle lengths of TDMA resources with fixed bandwidth, and it can handle real-time event-triggered loads with jitter and bursts. Since the method is based on evolutionary algorithms, it is however computationally expensive, and cannot guarantee a global optimal solution for a predetermined optimality criterion.

Contributions of this work:

- We present an analytic method to determine the provably smallest possible slot length that must be allocated in a TDMA resource with fixed cycle length and bandwidth, to serve a hard real-time load with arbitrary deterministic timing behavior.
- We present constructive methods to find the optimal cycle length and minimum required bandwidth of a TDMA resource.
- We show the applicability and computational efficiency of the presented methods in a case study of a large distributed embedded system, where we will find the optimal parameter set for a

0-7803-9451-8/06/$20.00

TDMA bus that interconnects 21 ECU's that send a total of 30 different hard real-time message streams with jitter and bursts over this shared communication resource.

- The presented work is based on an existing theoretical framework for modular system level performance analysis of hard real-time systems. We present a TDMA component that will extend this framework to enable performance analysis and interface-based design of distributed real-time systems with TDMA.

II. Modular Performance Analysis

In the domain of communication networks, powerful abstractions have been developed to model flow of data through a network. In particular Network Calculus [9] provides means to deterministically reason about timing properties of data flows in queuing networks. Real-Time Calculus [14] extends the basic concepts of Network Calculus to the domain of real-time embedded systems, and in [5] a unifying approach to Modular Performance Analysis with Real-Time Calculus has been proposed. It is based on a general event and resource model, allows for hierarchical scheduling and arbitration, and can take computation and communication resources into account.

The following sections introduce some concepts of Network and Real-Time Calculus, that build the foundation of this work. While we introduce these concepts from a communications point of view, all results can also be applied directly to the analysis and design of hardware/software components in a system. Then, messages correspond to tasks, and the message size to a task's execution time.

A. From Components to Abstract Components

In this work, we consider distributed embedded systems, consisting of a number of embedded computing units and a shared communication network. Every embedded computing unit (ECU) is connected to the communication network (CN) via a communication network interface (CNI), that uses the services of the communication network via a communication controller (CC). The ECU processes incoming event streams, and generates message streams that must be sent in real-time to other ECU's via the communication network. To initiate the sending of a message, the ECU places the message into one of possibly several FIFO buffers of the CNI. For real-time communication, every message stream has an associated maximum communication delay, that denotes the maximum time interval between the time a message of the stream is placed into the buffer, and the required delivery time of the message. The CNI uses the services of the CC to send the buffered messages over the network, and it applies an arbitration policy to share the available communication resource amongst the queued messages. Figure 2 shows on the left side a block diagram of the connection of an ECU to a communication network.

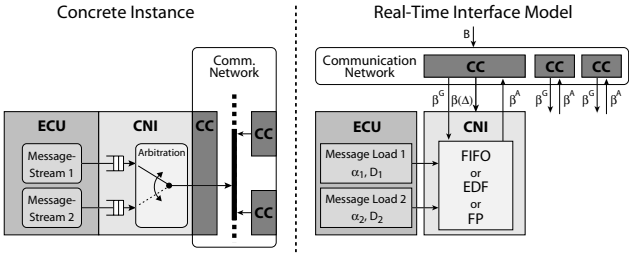

Fig. 2. A concrete ECU connected to a communication system (left), and the corresponding Real-Time Interface model (right).

We will use Real-Time Calculus [14, 5] to abstractly describe the real-time properties of such a CNI component. In comparison to the concrete CNI, an abstract CNI sends an abstract message stream over an abstract communication resource. In Real-Time Calculus, the timing and resource demand properties of the concrete message streams that enter the CNI are abstracted by Variability Characterization Curves (VCC) that are called arrival curves $\alpha(\Delta)$, following [9]. Together, $\alpha(\Delta)$ and the maximum allowable delay D of every message stream describe the properties of a message stream that are essential for real-time analysis, see [14]. The communication resource that enables the sending of the messages is modeled by a service curve $\beta(\Delta)$, see also [9]. $\beta(\Delta)$ is also a VCC, and describes the essential properties of the communication resources that are available to an abstract CNI component.

B. Variability Characterization Curves

An arrival curve $\alpha(\Delta) \in \mathbb{R}^{\geq 0}$, $\Delta \in \mathbb{R}^{\geq 0}$ provides an upper bound on the communication resource demand that messages arriving from an ECU in *any* time interval of length Δ create on a CNI, i.e. the messages that arrive on a stream within a time interval $[t, t+\Delta)$ may create a resource demand of at most $\alpha(\Delta)$ on an CNI, for all $t \geq 0$.

Arrival curves substantially generalize the classical representations of standard event arrival patterns such as sporadic, periodic, periodic with jitter or others. For example a message stream with messages of size e, that arrive with a period p, a jitter j, and a minimum inter arrival distance d, can be modeled by an arrival curve as follows:

$$\{p, j, d, e\} \quad \Rightarrow \quad \alpha(\Delta) = \min\left\{ \left\lceil \frac{\Delta + j}{p} \right\rceil e, \left\lceil \frac{\Delta}{d} \right\rceil e \right\} \qquad (1)$$

Besides being able to represent any message stream with known timing behavior, it is also possible to determine arrival curves corresponding to any finite length message trace, obtained for example from observation or simulation. Some examples of arrival curves are shown in Fig. 3.

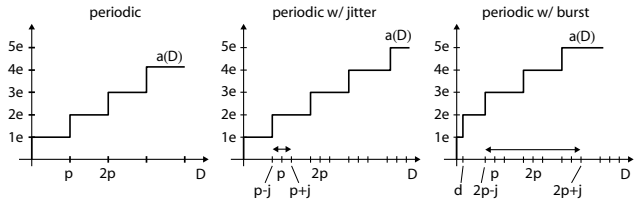

Fig. 3. Arrival curves for standard event arrival patterns.

Similarly, the resource availability of an abstract communication resource is modeled by a VCC called service curve. A service curve $\beta(\Delta) \in \mathbb{R}^{\geq 0}$, $\Delta \in \mathbb{R}^{\geq 0}$ provides a lower bound on the available resources, e.g. bus cycles, in any time interval Δ, i.e. in a time interval $[t, t+\Delta)$ at least $\beta(\Delta)$ bus cycles are available.

C. Real-Time Calculus

Network and Real-Time Calculus provide powerful methods to analyze different performance properties of systems, as for example detailed throughput or resource consumption analysis, see [9], [14] and [5]. Here, we however only consider delay and backlog analysis.

In Real-Time Calculus, the maximum *delay* Del_{max} experienced by messages of a message stream with arrival curve $\alpha(\Delta)$ that is sent over an abstract communication resource with service curve $\beta(\Delta)$ is bounded by the maximum horizontal distance between $\alpha(\Delta)$ and $\beta(\Delta)$:

$$Del_{max} \leq \sup_{\Delta \geq 0} \{\inf\{\tau \geq 0 : \alpha(\Delta) \leq \beta(\Delta + \tau)\}\} \qquad (2)$$

Analogously, the maximum *backlog* Buf_{max} of a message stream in the buffer of a CNI is bounded by the maximum vertical distance between $\alpha(\Delta)$ and $\beta(\Delta)$:

$$Buf_{max} \leq \sup_{\Delta \geq 0} \{\alpha(\Delta) - \beta(\Delta)\} \quad (3)$$

If a CNI serves more than one message stream, i.e. if it has more than one arrival curve as input, the total required buffer can be computed by replacing $\alpha(\Delta)$ with the sum of all arrival curves $\sum \alpha_i(\Delta)$ in (3). It is also possible to compute the different maximum delays, experienced by the message streams. For this, the arbitration policy of the CNI must be considered, see [5].

D. Real-Time Interfaces

Real-Time Interfaces were first introduced in [2], and they connect the principles of interface-based design [**?**] and Real-Time Calculus. The central idea of interface-based design is to describe components by a component interface that, if well-designed, provides enough information to decide whether a component can work properly in a system together with other components. Real-Time Interfaces can thereby be considered as a special instance of assume/guarantee interfaces, see also [**?**]. Components with an assume/guarantee interface make assumptions on the values of their input variables and give in return guarantees on the values of their output variables.

When we look at the connection of an ECU to a communication network in Fig. 2, then we see that the communication network provides a communication resource supply, represented as a service curve $\beta(\Delta)$, to a CNI. $\beta(\Delta)$ is therefore the output of an abstract communication network, and the input to an abstract CNI.

A communication network can therefore be modeled as an abstract component with a service output variable $\beta(\Delta)$, and its real-time interface would provide the output guarantee $\beta(\Delta) \geq \beta^G(\Delta)$ on this output variable. Through this output guarantee, the network component expresses that the service $\beta(\Delta)$, that is provided by the communication network is larger or equal $\beta^G(\Delta)$. $\beta^G(\Delta)$ is sometimes also referred to as supply bound function **sbf**(Δ), see e.g. [4] or [13].

An abstract CNI on the other hand can be modeled as a component with a service input variable $\beta(\Delta)$, and its real-time interface would make the input assumption $\beta(\Delta) \geq \beta^A(\Delta)$ on this input variable. Through this input assumption, the abstract CNI expresses that the service $\beta(\Delta)$, that is provided on its service input must be larger or equal $\beta^A(\Delta)$. In return, the abstract CNI then guarantees to send all messages within the required maximum delay. $\beta^A(\Delta)$ is sometimes also referred to as demand bound function **dbf**(Δ).

Figure 2 shows on the right side the Real-Time Interface model of an abstract communication network component with service guarantee $\beta^G(\Delta)$ that is connected to an abstract CNI component with service assumption $\beta^A(\Delta)$. In order to determine, whether the communication network provides enough service to the CNI, we only need to check that the following predicate is true:

$$\beta^G(\Delta) \geq \beta^A(\Delta) \quad \forall \Delta \geq 0 \quad (4)$$

The difficulty is then to find appropriate values for $\beta^G(\Delta)$ and $\beta^A(\Delta)$. The service assumption $\beta^A(\Delta)$ of an abstract CNI that serves a message stream with arrival curve $\alpha(\Delta)$ and maximum delay D is given by (see also Example 1):

$$\beta^A(\Delta) = \alpha(\Delta - D) \quad (5)$$

If on the other hand the CNI serves several message streams with an EDF arbitration policy, the service assumption can be computed as $\beta^A_{EDF}(\Delta) = \sum \alpha_i(\Delta - D_i)$, and for FIFO it can be computed as $\beta^A_{FIFO}(\Delta) = \sum \alpha_i(\Delta - D_{min})$. To compute the service assumption under a fixed priority arbitration policy is slightly more involved and is described in detail in [2], while [1] describes how to obtain the service assumption of components with mixed static and dynamic arbitration policies. $\beta^G(\Delta)$ on the other hand can be given as $\beta^G(\Delta) = B \cdot \Delta$ for a fully available communication network with bandwidth B, and in the next section, we will establish $\beta^G(\Delta)$ for a communication network with TDMA.

III. Performance Analysis of TDMA Systems

In this work, we analyze a real-time communication system consisting of n communication network interfaces that access via communication controllers a bus with bandwidth B that implements a TDMA protocol. The TDMA cycle length is denoted as $\bar{c}$ and can only take on values that are multiples of the cycle length quantum q_c. In every TDMA cycle, one single communication slot of length s_i is assigned to every CNI that is connected to the bus. In a realistic system, the slot lengths s_i can only take on values that are multiples of a slot length quantum q_s. We denote a quantized slot length as $\bar{s}_i = \lceil s_i/q_s \rceil \cdot q_s$. Further, every slot typically involves a slot overhead o_s, while the cycle itself involves a cycle overhead o_c. These overheads account for example for required network idle times between consecutive slots and cycles, CRC codes for channel fault detection, time synchronization data, or any other protocol related overhead. Depending on the different timing specifications, some bandwidth may remain unused in every communication cycle. Fig. 4 depicts the timing specifications in this TDMA protocol.

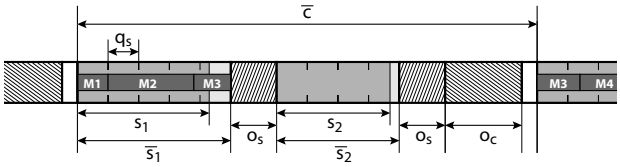

Fig. 4. TDMA protocol timing specifications.

In the CNI, a message service layer provides a packet service that fragments outgoing messages into packets and that reassembles incoming packets into messages. This message service layer guarantees that a CNI can always use the complete time slot assigned to it, as long as there are messages waiting to be sent. The overhead for the message fragmentation is accounted for in the slot overhead o_s. Since this fragmentation overhead depends on the message sizes and the arbitration policy of a CNI, one could introduce individual slot overheads $o_{s,i}$ for every CNI. The methods in this work could easily be extended accordingly.

A. TDMA Resource Component Interface

An abstract communication (or computation) resource with TDMA can be modeled by a real-time interface as depicted in Fig. 5. The component has an input B, that determines the total bandwidth of the underlying resource, and the cycle length of the TDMA protocol is specified by the parameter $\bar{c}$. The component further has n service outputs, that provide a service with the output guarantees $\beta_i(\Delta) \geq \beta^G_i(\Delta)$ to n connected CNI components with input assumptions $\beta^A_i(\Delta)$.

Internally, the service output guarantee $\beta^G_i(\Delta)$ can be determined by the slot length guarantee s^G_i that is assigned to the i^{th} CC. Analogously, we show in section IV, how the service input assumption $\beta^A_i(\Delta)$ of a connected CNI component can be transformed into a minimum slot length assumption s^A_i. The fulfillment of the communication service demand of a connected abstract CNI component can then be guaranteed directly by guaranteeing $s^G_i \geq s^A_i$.

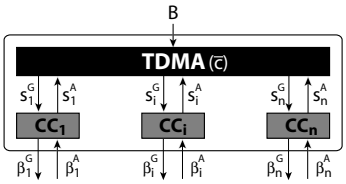

Fig. 5. Interface of an abstract resource component with TDMA.

B. Service Supply, Schedulability, Feasibility and Utilization

In a communication resource with TDMA, the i^{th} communication controller may not have access to the resource during a time interval that is limited by $\Delta = \bar{c} - s_i^G$. After this interval, the CC has exclusive access to the resource during a time interval of length s_i^G. A CC can therefore not guarantee any service to a connected CNI during any time interval $0 \leq \Delta < \bar{c} - s_i^G$, but it can guarantee a service of $B(\Delta - (\bar{c} - s_i^G))$ in any time interval $\bar{c} - s_i^G \leq \Delta < \bar{c}$. This service guarantee can be expressed as

$$\beta_i^G(\Delta) = B \ \max\left(\left\lfloor \frac{\Delta}{\bar{c}} \right\rfloor s_i^G, \Delta - \left\lceil \frac{\Delta}{\bar{c}} \right\rceil (\bar{c} - s_i^G)\right) \quad (6)$$

or more compactly as

$$\beta_i^G(\Delta) = B \sup_{0 \leq \lambda \leq \Delta} \left\{ \lambda - \left\lceil \frac{\lambda}{\bar{c}} \right\rceil (\bar{c} - s_i^G) \right\} \quad (7)$$

We define that a real-time communication system is said to be *schedulable*, if the connected CNI's can fulfill the real-time requirements of all message streams, i.e. if all messages in the system can be delivered within an time interval that is limited by their maximum allowable delay D. According to the theory of Real-Time Interfaces, a real-time communication system is therefore schedulable if

$$\beta_i^G(\Delta) \geq \beta_i^A(\Delta) \quad \forall i, \forall \Delta \geq 0 \quad (8)$$

or equivalently if $s_i^G \geq s_i^A \quad \forall i$.

We further define that a real-time communication system with TDMA is *feasible*, if the sum of the required slot lengths and the protocol overhead is less or equal the cycle length, i.e. if

$$\bar{c} \geq \sum_{\forall i} s_i^A + o_c + no_s \quad (9)$$

Following this definition of feasibility, the slot length guarantee s_i^G to the i^{th} CC can then be computed as

$$s_i^G = \bar{c} - \left(\sum_{\forall j \neq i} s_j^A + o_c + no_s \right) \quad (10)$$

We define the utilization σ_i^A as the quotient of the slot length s_i^A divided by the cycle length $\bar{c}$. Analogously to (9), a TDMA system is then feasible if the total utilization is less or equal one:

$$1 \geq \sum_{\forall i} \sigma_i^A + \frac{o_c + no_s}{\bar{c}} \stackrel{def}{=} \sigma_{tot}(\bar{c}) \quad (11)$$

C. Delay and Backlog

For delay and backlog analysis in a communication system with TDMA as defined above, we may now use (2) and (3). We only need to replace $\beta(\Delta)$ with $\beta_i^G(\Delta)$ in the corresponding formulas.

EXAMPLE 1. *Suppose we have a message stream M_0 with $p_0 = 198ms$, $j_0 = 387ms$ and $d_0 = 48ms$. The single messages have a size of 12 units, and a maximum allowable delivery delay of $D_0 = 110ms$. The messages are sent over a communication resource with TDMA, that has a total bandwidth of $B = 1'000units/s$. The TDMA cycle has a length of $\bar{c} = 80ms$, and a slot of length $s_0^G = 20ms$ is assigned to the CNI that sends the message stream M_0.*

Figure 6 shows the arrival curve α_0, the service assumption β_0^A, as well as the service guarantee β_0^G of the above system, computed according to (1), (5) and (7), respectively. Since $\beta_0^G \geq \beta_0^A$, we know that the real-time requirements of the message stream are fulfilled, following (8). And using (2) and (3), we can compute the maximum delay experienced by a message as $Del_{max,0} = 96ms$, and the maximum backlog in the CNI as $Buf_{max,0} = 24units$.

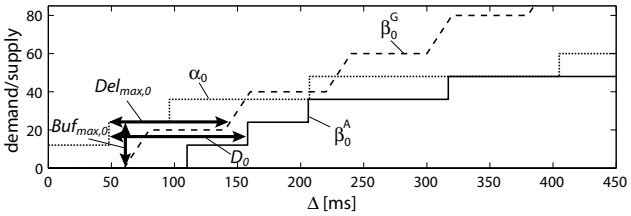

Fig. 6. Arrival and service curves of Example 1.

IV. MINIMUM SLOT TIME ALLOCATION

Based on the powerful abstractions of service guarantee and service assumption curves, and the explicit schedulability requirement (8) coming from the theory of Real-Time Interfaces, it is possible to determine the exact minimum time slot s_i^A that must be assigned to a CNI with service assumption $\beta_i^A(\Delta)$ to be schedulable on a TDMA communication network with bandwidth B and cycle length $\bar{c}$. For this, we need to construct the inverse of (6) with respect to s_i^G as the smallest s_i^G that leads to a service supply that fulfills the schedulability requirement (8):

$$s_i^A = \sup_{\Delta \geq 0} \left\{ \min\left(\frac{\beta_i^A}{B \left\lfloor \frac{\Delta}{\bar{c}} \right\rfloor}, \frac{\beta_i^A - B\Delta + B \left\lceil \frac{\Delta}{\bar{c}} \right\rceil \bar{c}}{B \left\lceil \frac{\Delta}{\bar{c}} \right\rceil} \right) \right\} \quad (12)$$

This minimum time slot s_i^A is provably the smallest possible time slot allocation that guarantees a service supply $\beta_i^G(\Delta) \geq \beta_i^A(\Delta)$ on a TDMA resource with bandwidth B and slot length $\bar{c}$.

EXAMPLE 2. *Figure 7 shows the minimum slot length $s_0^A(\bar{c})$ of message stream M_0 from Example 1, as a function of the TDMA cycle length $\bar{c}$. We see that the minimum required slot length increases with increasing cycle length, and it is lower bounded by $s_0^A(\bar{c}) \geq \bar{c} - (D_0 - e_0)$, because the gap between two consecutive slots must never be greater than $D_0 - e_0$.*

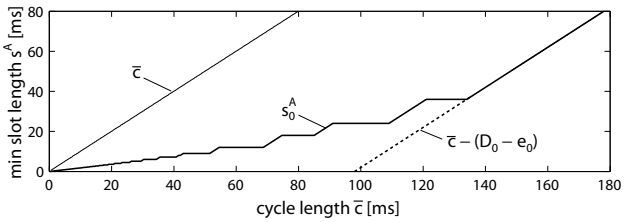

Fig. 7. Minimum slot lengths for message stream M_1.

Figure 8 shows the minimum service guarantees $\beta_0^G(\Delta)$ for three different cycle lengths, computed by setting $s_0^G(\bar{c}) = s_0^A(\bar{c})$.

Using (12), we can now compute the minimum slot lengths for all CNI's in a communication network with bandwidth B and TDMA

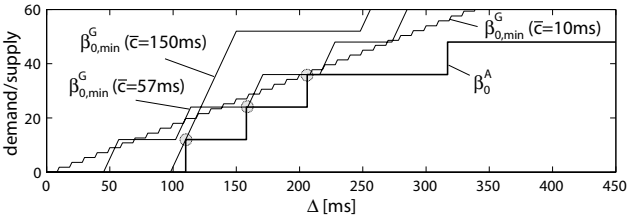

Fig. 8. Minimum service guarantees for different cycle lengths.

cycle length $\bar{c}$. If a slot allocation with these minimum slot lengths leads to a feasible communication system according to (9), i.e. if the sum of the minimum slot lengths plus the protocol overhead is smaller than the cycle length, then we can use $\bar{c}$ and $s_i = s_i^A(\bar{c})$ as TDMA settings. Otherwise we are guaranteed that no feasible slot allocation exists for the cycle length $\bar{c}$.

EXAMPLE 3. *Consider a communication network with* 10 *CNI's, each one serving one of the message streams M_0 to M_9 specified in Table I. This specification equals the specification of System 3 in [7]. According to [7], this was the most difficult system to optimize in [7].*

TABLE I– EXAMPLE SYSTEM WITH 10 MESSAGE STREAMS.

	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9
p	198	102	283	354	239	194	148	114	313	119
j	387	70	269	387	222	260	91	13	302	187
d	48	45	58	17	65	32	78	-	86	89
e	12	7	7	11	8	5	13	14	5	6
D	110	140	115	145	180	140	200	120	140	100

To find feasible TDMA parameters for this system, we computed $\sigma_{tot}(\bar{c})$ for $\bar{c} \in [0.1ms \ldots 600ms]$ with a cycle quantum $q_c = 100\mu s$. In [7] no slot length quantization or protocol overhead was considered, Fig. 9 depicts the corresponding results. Additionally, we computed $\bar{\sigma}_{tot}(\bar{c})$ with a slot length quantum $q_s = 10\mu s$, but still without protocol overhead. The results are also shown in Fig. 9.

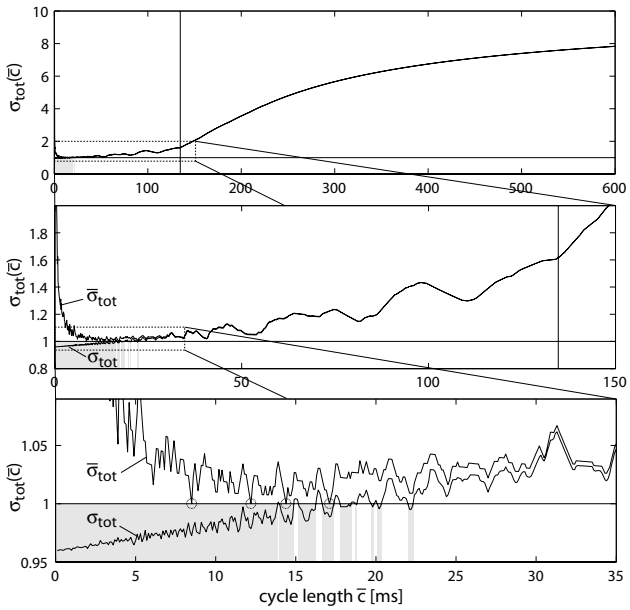

Fig. 9. Total utilization as a function of the TDMA cycle length.

Remember, that feasible TDMA parameters exists for a cycle length $\bar{c}$ if, and only if $\sigma_{tot}(\bar{c}) \leq 1$. Without considering quantization, this is the case for all values in the grey shaded areas in Fig. 9. When we consider a small slot quantization however, only the four encircled values of $\bar{c}$ still lead to feasible TDMA parameters.

In general, if we do not consider quantization effects and protocol overhead, then the smallest possible $\bar{c}$ will always lead to feasible TDMA parameters, if the total bandwidth B is large enough. As soon as we consider quantization effects and protocol overhead however, arbitrary small values for $\bar{c}$ are not feasible anymore.

But also arbitrary large values for $\bar{c}$ will not lead to feasible TDMA parameters, because the slot lengths are lower bounded by $s_0^A(\bar{c}) \geq \bar{c} - (D_0 - e_0)$. $\sigma_{tot}(\bar{c})$ will therefore strive towards the number of CNI's in the system for large $\bar{c}$, i.e. towards 10 in Example 3.

Feasible TDMA parameters will therefore only exist for cycle lengths $\bar{c}$ that are not too small and not to large. And as we can be seen in Fig. 9, the total utilization $\sigma_{tot}(\bar{c})$ as a function of the TDMA cycle length has a very complex and nonlinear behavior. Often, intervals of feasible cycle lengths are even non-contiguous.

V. OPTIMAL CYCLE LENGTH

In a typical TDMA system, not only the slot lengths s_i, but also the cycle length $\bar{c}$ is a customizable parameter. To find an optimal cycle length for a TDMA system, we first need to define an optimality criterion. One possible optimality criterion that we will use in this work is the average remaining bandwidth $\sigma_r = 1 - \sigma_{tot}$. This remaining bandwidth could be distributed additionally to the existing CNI's, or it could be used to admit additional future load in a dynamic system.

Note, that if we want to account for up to m dynamically added CNI's in a dynamic system, we need to consider the slot overheads for these dynamically added CNI's. Further, for systems with a slot quantum, we need to consider that only multiples of full slot quantums can be assigned to existing or future CNI's:

$$\bar{\sigma}_r(\bar{c}) = \left\lfloor \left(1 - \sum_{\forall i} \bar{\sigma}_i(\bar{c}) - \frac{(n+m)o_s + o_c}{\bar{c}} \right) \frac{\bar{c}}{q_s} \right\rfloor \frac{q_s}{\bar{c}} \qquad (13)$$

In Fig. 9, we have seen that the total utilization $\sigma_{tot}(\bar{c})$ as a function of the TDMA cycle length has a very complex and nonlinear behavior. To find the optimal cycle length with the maximum remaining bandwidth σ_r, we therefore have no choice but to compute σ_r for all possible values of $\bar{c}$.

However, in Example 2 and in Fig. 7, we have seen that the required slot lengths are lower bounded by $s_1^A(\bar{c}) \geq \bar{c} - (D_1 - e_1)$. It is therefore possible to find an upper bound to feasible cycle lengths:

$$\bar{c}_{max} = \sup_{\bar{c} \geq 0} \left\{ \bar{c} \ : \ \bar{c} \geq \sum_{\forall i} \max(0, \bar{c} - (D_i - e_i)) \right\} \qquad (14)$$

Due to this upper bound, σ_r needs only to be computed for $\bar{c}_{max}/q_c$ different values. The upper bound for the system in Example 3 is $\bar{c}_{max} = 134,7ms$ and is indicated by the vertical bar in Fig. 9.

VI. MINIMUM TOTAL SERVICE BANDWIDTH

At design time, the service bandwidth B is often also a customizable parameter. The minimum total service bandwidth B_{min} is the smallest possible service bandwidth B of a TDMA system with service assumptions β_i^A, for which feasible slot allocations s_i exists:

$$B_{min} = \inf_{B \geq 0} \{ B \ : \ \sigma_{tot,min}(B) \leq 1 \} \qquad (15)$$

with

$$\sigma_{tot,min}(B) = \inf_{\bar{c} \leq \bar{c}_{max}} \{ \sigma_{tot}(\bar{c}, B) \} \qquad (16)$$

From (12), it can be seen that the minimum slot length s_i^A is monotonically decreasing with increasing service bandwidth B, i.e. $s_i^A(B + dB) \leq s_i^A(B)$. Because of this, the total utilization σ_{tot} is also monotonically decreasing, and as a consequence also the minimum total utilization $\sigma_{tot,min}$ is monotonically decreasing with increasing service bandwidth B, i.e. $\sigma_{tot,min}(B+dB) \leq \sigma_{tot,min}(B)$. We can therefore find the minimum total service bandwidth B_{min} (or B^A in Real-Time Interface notation) by using binary search.

VII. Case Study

The case study systems consists of 21 ECU's that are interconnected with a TDMA bus, and that send a total of 30 different hard real-time message streams with jitter and bursts that are specified in Table II. The TDMA bus has a cycle quantum $q_c = 1ms$, a slot quantum $q_s = 0.5ms$, a slot overhead $o_s = 0.5ms$, and a cycle overhead $o_c = 2.5ms$. On ECU_0 and ECU_1, EDF is used to arbitrate the sending of $M_0 - M_3$ and $M_4 - M_6$, FIFO is used on ECU_2 to arbitrate $M_7 - M_9$, and FP is used on ECU_3 to arbitrate $M_{10} - M_{12}$. The remaining 17 ECU's send only a single message stream each.

TABLE II– Case study system with 30 message streams.

	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9
p	196	245	105	147	231	308	275	234	273	182
j	387	70	269	387	222	260	91	387	70	269
d	48	-	58	17	65	-	-	48	-	58
e	7	4	6	1	8	2	3	5	5	3
D	176	237	115	488	206	311	275	207	178	198
	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19
p	153	357	476	302	258	424	287	451	539	309
j	80	70	177	967	719	257	38	159	11	153
d	-	-	-	27	89	-	-	-	-	-
e	2	2	3	2	2	4	7	6	13	11
D	153	423	556	511	371	315	210	245	196	413
	M20	M21	M22	M23	M24	M25	M26	M27	M28	M29
p	506	357	304	510	298	243	457	502	247	226
j	250	393	278	296	184	400	300	312	365	278
d	-	3	40	-	-	18	-	-	83	85
e	4	3	4	5	4	5	7	3	8	7
D	245	336	378	126	161	469	574	560	133	301

We first search the minimum required total service bandwidth for the TDMA bus in this system. From this we learn, that a minimum bandwidth of $B_{min} = 1.27Mbit/s$ is required. With this bandwidth, feasible TDMA settings exist for a cycle length $\bar{c} = 92ms$, and lead to a total utilization of $\bar{\sigma}_{tot} = 1$.

In a next step, we choose a TDMA bus with total bandwidth of $B = 1.5Mbit/s \geq B_{min}$. We want to optimize slot and cycle lengths on this bus according to section V, such that 5 additional ECU's could be added at a later point of time. For this we compute (13) up to $\bar{c}_{max} = 169ms$. The results are shown in Fig. 10 and suggest to use a cycle length of again $\bar{c} = 92ms$. This leads to a maximum remaining average bandwidth of $\bar{\sigma}_r = 0.11$.

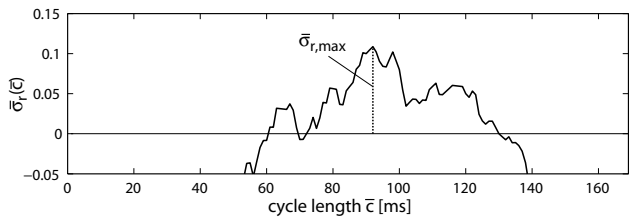

Fig. 10. Remaining average bandwidth in the case study system.

VIII. Computational Complexity

To analyze the examples and case study in this paper, we used a prototype implementation of Real-Time Calculus and Real-Time Interfaces that is implemented in Java and uses Matlab as user frontend. We run this prototype tool on a Pentium Mobile 1.6 GHz.

Computing Fig. 9 took $7.5s$, while computing the same results up to the required upper bound $\bar{c}_{max}$ took only $1.1s$. In the case study, finding the minimum total bandwidth and the corresponding cycle and slot lengths took $0.31s$ per iteration, and optimizing the cycle and slot lengths for the TDMA bus with fixed bandwidth took also $0.31s$.

When computation needs to be faster (e.g. for exploration), linear approximated VCC's could be used, that trade off computational complexity with the tightness of the results, see e.g. [3].

IX. Conclusions

We presented an analytic method to determine the provably smallest possible slot length that must be allocated in a TDMA resource, to serve an event-triggered hard real-time load. We further presented constructive methods to find the optimal cycle length as well as the minimum required bandwidth of a TDMA resource. Using these new methods, it is now possible to determine the minimum required bandwidth, as well as the optimal slot and cycle parameters for a TDMA resource, when we are initially only given a set of real-time load specifications that must be served by the TDMA resource. The applicability and computational efficiency of the presented methods was shown in a case study, where finding the optimum TDMA parameter set for a large distributed embedded system with 21 ECU's that send a total of 30 different hard real-time message streams took less than a second. Finally, we extended an existing theoretical framework for modular system level performance analysis of hard real-time systems by introducing a component that models a TDMA resource. This component enables to use the framework for performance analysis and interface-based design of complete distributed real-time systems with TDMA.

References

[1] Interface-Based Design of Real-Time Systems with Hierarchical Static and Dynamic Scheduling. Authors omitted for blind review.

[2] Real-Time Interfaces for Interface-Based Design of Real-Time Systems with Fixed Priority Scheduling. Authors omitted for blind review.

[3] K. Albers and F. Slomka. An event stream driven approximation for the analysis of real-time systems. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS'04)*. IEEE Press, 2004.

[4] S. K. Baruah. Dynamic- and static-priority scheduling of recurring real-time tasks. *Real-Time Systems*, 24(1):93–128, 2003.

[5] S. Chakraborty, S. Künzli, and L. Thiele. A general framework for analysing system properties in platform-based embedded system designs. In *Proc. 6th Design, Automation and Test in Europe (DATE)*, pages 190–195, March 2003.

[6] K. Goossens, J. Dielissen, J. van Meerbergen, P. Poplavko, A. Rădulescu, E. Rijpkema, E. Waterlander, and P. Wielage. Guaranteeing the quality of services in networks on chip. In *Networks on chip*, pages 61–82. Kluwer Academic Publishers, Hingham, MA, USA, 2003.

[7] A. Hamann and R. Ernst. TDMA Time Slot and Turn Optimization with Evolutionary Search Techniques. In *Design, Automation and Test in Europe (DATE 2005)*, 2005.

[8] H. Kopetz. *Real-Time Systems - Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997.

[9] J. Le Boudec and P. Thiran. *Network Calculus - A Theory of Deterministic Queuing Systems for the Internet*. LNCS 2050, Springer Verlag, 2001.

[10] R. Obermaisser. CAN Emulation in a Time-Triggered Environment. In *Proceedings of the 2002 IEEE International Symposium on Industrial Electronics (ISIE)*. IEEE, 2002.

[11] R. Obermaisser. *Event-Triggered and Time-Triggered Control Paradigms*, volume 22 of *Real-Time Systems Series*. Springer, 2005.

[12] P. Pop, P. Eles, Z. Peng, V. Izosimov, M. Hellring, and O. Bridal. Design Optimization of Multi-Cluster Embedded Systems for Real-Time Applications. In *Design, Automation and Test in Europe (DATE 2004)*, pages 1028–1033, 2004.

[13] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In *Proceedings of the Real-Time Systems Symposium (RTSS)*, pages 2–13. IEEE Press, 2003.

[14] L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 4, pages 101–104, 2000.