On the Quantification of Sustainability and Extensibility of FlexRay Schedules

Reinhard Schneider, Dip Goswami, Samarjit Chakraborty
TU Munich, Germany

Unmesh Bordoloi, Petru Eles, Zebo Peng
Linkoeping University, Sweden

ABSTRACT

FlexRay has emerged as the de-facto next generation in-vehicle communication protocol. Messages are scheduled incrementally on FlexRay according to the automotive design paradigm where new applications are added iteratively. On this account, the schedules must be (i) *sustainable*, i.e., when messages are added in later iterations, they must preserve deadline guarantees of existing messages and (ii) *extensible*, i.e., they must accommodate future messages without changes to existing schedules. Unfortunately, traditionally used metrics of sustainability and extensibility for timing and schedulability analysis are generic and can not be trivially adapted to FlexRay schedules. This is because of platform-specific properties of FlexRay like being a hybrid paradigm, where both time-triggered and event-triggered segments are used for communication. In this paper, we first introduce new notions of sustainability and extensibility for FlexRay that capture protocol-specific properties and then present novel metrics to quantify sustainable and extensible schedules. We demonstrate the applicability of our results with industrial-size case studies and show that our proposed metrics may be visually represented allowing easy interpretation by system designers in the automotive industry.

Categories and Subject Descriptors

C.3 [**Special-purpose and application-based systems**]: Real-time and embedded systems

General Terms

Performance, Design

Keywords

Automotive, FlexRay, Scheduling, Real-time systems

1. INTRODUCTION

FlexRay has taken a veritable lead as the next generation in-vehicle communication network. The FlexRay protocol is developed by a consortium of more than 100 leading companies of the automotive industry [3] and already has been deployed in vehicles from Audi and BMW. The design process for FlexRay follows the iterative design paradigm of the automotive industry, where new components are added at each iteration incrementally. Thus, new messages are added and scheduled on the FlexRay bus at each design cycle. Fig. 1 illustrates an overview of this incremental design process. At the initial design cycle, the system designer decides on the physical layer configuration, e.g., the bus topology and configures global parameters, such as bus speed and bus period. Once these parameters are validated and fixed they are not altered in later iterations in order to avoid a complete re-design and re-evaluation of the system from scratch, which is extremely costly. For the same reason, it is required that messages can be added in future iterations without disturbing the schedules of the existing messages. Thus, schedules generated at each iteration should be (i) *sustainable*, i.e., future messages should not violate any message deadlines and (ii) *extensible*, i.e., future messages can be accommodated.

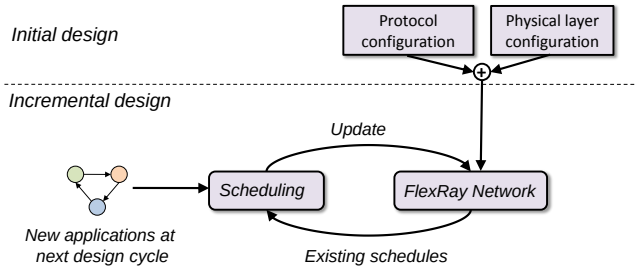

Figure 1: Overview of the design process.

Our contributions and related work: While there have been efforts to capture the sustainability and extensibility of schedules in real-time systems, the derived metrics can not be applied to FlexRay. For example, sustainability was discussed in the context of real-time systems in [2] and various papers focused on extensibility of distributed systems [6, 11]. However, these metrics are not applicable to domain-specific protocols like FlexRay because of its special properties like the hybrid paradigm (FlexRay has both time-triggered and event-triggered segments) and slot-multiplexing.

Apart from the above line of work, of late, there has also been tremendous research interest towards building tools and algorithms for timing analysis and scheduling of messages on FlexRay-based automotive networks. A technique to schedule messages on the static (time-triggered) segment of FlexRay was described in [5], while a timing analysis method for the dynamic (event-triggered) segment of FlexRay was proposed in [7]. The work in [4, 10] attempted to schedule messages incrementally or to incorporate uncertainty of design parameters. However, no well-defined metrics that quantify the sustainability and extensibility of the schedules on FlexRay were presented. Moreover, two major simplifying assumptions were made. First, the proposed design frameworks do not capture the spirit of FlexRay-specific properties like slot multiplexing. Secondly, they were restricted to the static segment.

In this paper, we propose definitions for both sustainability and extensibility from the perspective of the FlexRay protocol-specific details like slot-multiplexing, and also present metrics in order to quantify the quality of FlexRay schedules. Towards this, metrics

DAC 2011, June 5–10, 2011, San Diego, California, USA.

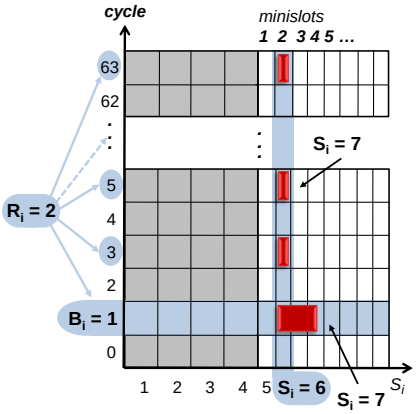

Figure 2: The FlexRay 64-cycle matrix.

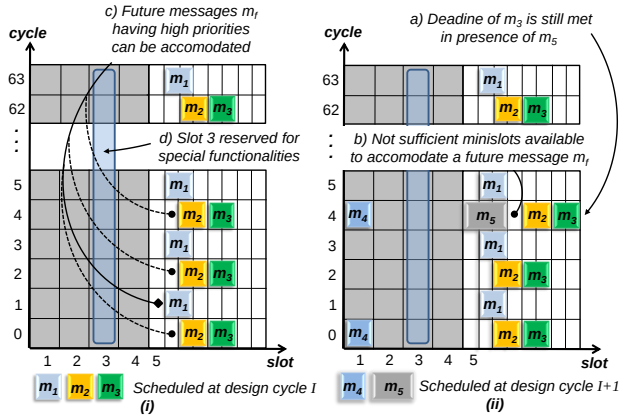

Figure 3: Motivational examples.

proposed here are applicable to both, static and dynamic segments. Finally, apart from having a mathematical basis, our metrics allow easy visualization and thus, they may be easily interpreted by engineers from the automotive domain. We would like to highlight that our focus here is on developing new FlexRay-specific metrics for quantifying the extensibility and sustainability of schedules. In this paper, we do not address the problem of *synthesizing* schedules based on these metrics in any detail. We envisage that our work will pave the way for designing new scheduling algorithms – based on the proposed metrics – along the lines of [5, 7, 8, 10].

2. THE FLEXRAY PROTOCOL

The FlexRay Communication protocol [3] is organized as a periodic sequence of cycles, where each cycle is of fixed length *gdCycle*. Each cycle is further subdivided into two major segments: static (ST) and dynamic (DYN). In the following we discuss the ST and the DYN segments, followed by specific properties of the 64-cycle matrix which represents a periodic sequence of bus cycles.

Static segment: The ST segment of FlexRay follows the time-triggered communication paradigm. Thus, it is partitioned into equal-length time windows, called *slots*. The set of static slots is denoted by $\mathcal{S}_{ST} = \{1, ..., S_{lS}\}$. Each message m_i to be transmitted in the ST segment is assigned a *slot number* $S_i \in \mathcal{S}_{ST}$. If m_i is not ready at the beginning of the slot, the slot remains empty.

Dynamic segment: The set of dynamic slots is given by $\mathcal{S}_{DYN} = \{S_{fD}, ..., S_{lD}\}$. A dynamic *slot* is a logical entity, which rather specifies the *priority* of a message in the DYN segment. Thus, each message m_i is assigned a slot number $S_i \in \mathcal{S}_{DYN}$, which specifies that m_i may be transmitted at the beginning of slot S_i. Messages having a higher priority are assigned lower slot numbers so that they have access to the bus first. Further, the slots in the DYN segment comprise of a number of equal-length *minislots*. In case a message m_i is transmitted in a slot $S_i \in \mathcal{S}_{DYN}$, then this slot consumes a certain number of minislots depending on the message size. However, if no message is transmitted in $S_i \in \mathcal{S}_{DYN}$, then only one minislot is consumed. For example, in cycle 1 of Fig. 2, a message transmission starts in minislot 2 and occupies 3 successive minislots. Thus, after message transmission the minislot count changes from 2 to 5 while the slot count changes from 6 to 7. Note, that when its turn comes, a message is transmitted only if the current minislot counter value is not greater than *pLatestTx* which denotes the highest minislot counter value a message transmission is allowed to begin for a certain Electronic Control Unit (ECU). The value of *pLatestTx* is statically configured during design time and depends on the maximum dynamic payload size that is allowed to be transmitted by a certain ECU (please see [3]).

FlexRay schedules: In the above we described the ST and the DYN segments of the FlexRay communication cycle. However, a set of 64 such communication cycles is repeated in a periodic sequence where each cycle is indexed by a cycle counter. The cycle counter is incremented from 0 to 63 after which it is reset to 0. Apart from the slot number S_i two further parameters specify the actual transmission cycles within the 64 cycles: (i) the base cycle B_i indicating the offset within 64 communication cycles, and (ii) the cycle repetition rate R_i, which denotes the number of cycles that must elapse between two consecutive allowable transmissions. Thus, any FlexRay message m_i is assigned S_i, B_i, and R_i to uniquely specify admissible transmission points within 64 cycles which we refer to as the *schedule* $\Theta_i = \{S_i, B_i, R_i\}$ of m_i. For instance, the schedule depicted in Fig. 2 is specified by $\Theta_i = \{6, 1, 2\}$, i.e., every odd cycle in slot 6 is available for transmission of message m_i. This is because the first cycle is indicated by $B_i = 1$ and $R_i = 2$ specifies that two cycles must elapse between allowable transmission points. Note that, every even cycle in slot 6 may be assigned to a different message, e.g., $\Theta_j = \{6, 0, 2\}$. Such scheduling leads to *slot-multiplexing*, i.e., the same slot is being used by multiple messages in different cycles. Since any message will be scheduled within the 64 cycles, the base cycle can be assigned a value within 0 and 63, i.e., $B_i \in \{0, ..., 63\}$. According to the FlexRay protocol ([1], [3]), the relations (i) $R_i = 2^r$; $r \in \{0, ..., 6\}$ and (ii) $B_i < R_i$ must hold among these parameters for any schedule Θ_i. Further, we refer to the set $\Gamma_i := \{\gamma \mid \gamma = (B_i + n \cdot R_i) \mod 64,\ n \in \mathbb{N}_0\}$ as the set of feasible cycles of Θ_i.

3. MOTIVATION AND CHALLENGES

In this section, we illustrate the challenges involved in quantifying sustainability and extensibility from the perspective of the FlexRay protocol discussed in the previous section. Towards this, we show that conventional metrics do not apply to FlexRay and motivate the need for novel techniques that quantify sustainable and extensible schedules considering platform-specific properties. Let us consider the examples in Fig. 3 that illustrate several schedules in two consecutive design cycles I and $I+1$. The figure shows three messages m_1, m_2, and m_3 that have been scheduled at design cycle I on the DYN segment and message m_4 and m_5 that were scheduled on ST and DYN segment, respectively, at iteration $I + 1$. Note that messages in the ST segment do not experience interference from future messages as static slots are of fixed and equal length. Consequently, a schedule Θ_i in the ST segment is sustainable if the deadline d_i is guaranteed at schedule synthesis. Hence, we will mostly focus on the DYN segment.

Example (a): At iteration $I + 1$ (Fig. 3(ii)), message m_5 has been assigned slot 5 which has the highest priority in the DYN segment. Note that at iteration I, m_3 was already assigned slot 8 in cycle 4. Hence, the schedule of m_3 must be designed such that its deadline d_3 is met even in the presence of the new message m_5, i.e., the

schedule of m_3 must be *sustainable*. Towards this, two pieces of information are crucial during schedule synthesis at design cycle I, (i) the number of future messages having higher priorities than m_3 and (ii) the worst-case workload generated by the future messages. Clearly, designers can not precisely predict such information. However, based on the product line and class of applications expected in future, it is reasonable to assume some knowledge about the range of typical message sizes. Given such a range, we will show in Section 4 how protocol properties allow to predict the worst-case workload in future with reasonable accuracy. Moreover, given a schedule Θ_i, we may bound the number of higher priority messages that may be assigned according to the slot number $S_i \in \Theta_i$. Based on such observations, we will specify a notion of sustainability and present a sustainability test in Section 4.

Example (b): Note that m_3 is only allowed to be transmitted in the DYN segment if there are sufficient minislots available, i.e., m_3 can only be transmitted if the minislot count does not exceed *pLatestTx* (see Section 2). Let a future message m_f consume 2 minislots and *pLatestTx*=7. Then, as depicted in Fig. 3(ii), m_f will not be allowed to allocate cycle 4 of slot 6 as there will not be sufficiently many minislots available for a transmission of m_3 before *pLatestTx*. Hence, schedules such as $\Theta_f = \{6, 0, 2\}$, $\Theta_f = \{6, 0, 4\}$, $\Theta_f = \{6, 4, 8\}$, etc., should not be assigned to m_f. Our metric for defining sustainability must be able to capture such details as well.

Example (c): In this example we show that conventional metrics like counting the total number of empty slots do not accurately quantify extensibility for the FlexRay protocol. For example, slot 6 is assigned to message m_1 in every odd cycle at iteration I. Conventional metrics such as counting the total number of empty or occupied slots do not account for the schedule properties discussed in Section 2 because a future message m_f may be assigned slot 6 in the even cycles using slot-multiplexing. Hence, the schedules $\Theta_f = \{S_f, B_f, R_f\}$ where $S_f = 6$, $R_f \in \{2, 4, 8, 16, 32, 64\}$, and $B_f = 2n < R_f, n \in \mathbb{N}_0$, are available for m_f. In order to quantify such available schedules accurately, we require novel metrics of extensibility which we will present in Section 5.

Example (d): System designers often reserve certain slots that are provided for specific protocols, e.g., XCP which is a measurement and calibration protocol, or protocols for diagnosis and transport layer. Such slots may not be assigned to application messages even if they are empty. For instance, in Fig. 3(i), slot 3 is reserved for such special functionalities. Hence, future application messages that must be scheduled in the ST segment may not be assigned to any cycles of slot 3, e.g., m_4 has been scheduled in the first (and not the third) slot at iteration $I + 1$ (see Fig. 3(ii)). Consequently, we do not quantify the extensibility of reserved slots in order to avoid distortion of the extensibility metric (see Section 5).

4. SUSTAINABILITY ANALYSIS.

In this section we define the notion of *sustainability* for FlexRay schedules. A schedule Θ_i is called *sustainable* iff

- the deadline d_i is guaranteed even in the presence of future messages interfering with m_i
- the size of future messages interfering with m_i do not exceed c_x minislots where c_x is the average resource consumption
- existing schedules are not changed at any time.

As mentioned in Section 3, the quantification of sustainability for DYN segment schedules is more complex due to the dynamic nature of the priority-based communication paradigm. In the following, we first present (i) a *delay model* to compute message delays, and (ii) a *workload estimation model* to account for interference due to future messages. In general, the number and workload of future higher priority messages is unpredictable. However, we exploit certain FlexRay-specific properties to bound the number of higher priority messages and make reasonable assumptions on the precipitated workload. Finally, we present a *sustainability test* to check if any schedule Θ_i satisfies real-time constraints in the presence of future messages interfering with m_i.

Delay model: The transmission time of a message m_i consuming c_i minislots is given by $e_i = c_i \cdot t_{MS}$ where t_{MS} is the duration of a minislot. As one minislot is consumed even if no message is transmitted in its assigned slot $S_i \in \mathcal{S}_{DYN}$, we denote the effective transmission time by $\overline{e}_i = (c_i - 1) \cdot t_{MS}$. This captures the additional transmission time in case a message m_i is transmitted on the bus. Let $\mathcal{G}_k$ be the sets of message indices $j \in \mathcal{G}_k$ such that $\Gamma_j \cap \gamma_k \neq \emptyset$ and $S_j < S_i$, $\forall \gamma_k \in \Gamma_i$. Hence, m_j are messages having a priority higher than m_i, i.e., they share at least one cycle with m_i and have a lower slot number such that their transmissions might affect the delay of m_i. Provided that a sufficiently large number of minislots is available to transmit m_i in any cycle, the worst-case delay due to messages having a higher priority than m_i is computed as

$$D_i = R_i \cdot gdCycle + \max_{\forall k} \sum_{j \in \mathcal{G}_k} \overline{e}_j + e_i. \tag{1}$$

The first term in (1) accounts for the bus blocking time ($R_i \cdot gdCycle$) in case m_i just missed its slot S_i and has to wait for R_i cycles until the next available slot. The second component $\max_{\forall k} \sum_{j \in \mathcal{G}_k} \overline{e}_j$ captures the worst-case interference due to messages m_j having a higher priority than m_i and e_i denotes the transmission time of m_i. Further, the number of future messages with higher priorities than m_i corresponding to $\mathcal{G}_k$, may be bounded by

$$x_k = S_i - S_{lS} - |\mathcal{G}_k| - 1 \tag{2}$$

where $|\mathcal{X}|$ denotes the cardinality of set $\mathcal{X}$. However, the transmission time $\overline{e}_j$ of the future messages needs to be estimated which will be discussed in what follows.

Workload estimation model: Let $\mathcal{N}$ be the set of feasible payload sizes $n \in \{0, 2, 4, ..., 254\}$ in bytes in accordance with the FlexRay protocol. Let $\mathcal{C}$ be the set of message sizes c in terms of minislots, including protocol header and physical layer properties. Then, $f : \mathcal{N} \rightarrow \mathcal{C}$ is defined according to [3] as

$$c = f(n) = 1 + \left\lceil \frac{g(n) + h_1(\theta)}{h_2(\theta)} \right\rceil + h_3(\theta) \tag{3}$$

where the functions $h_1(\theta), h_2(\theta)$, and $h_3(\theta)$ denote functions of FlexRay network parameters θ, and $g(n)$ accounts for the frame size including the frame header as a function of the payload size n. We omit the details on these parameter relationships due to space constraints. Please refer to [3] for a detailed discussion. Effectively, there exist subsets $\mathcal{N}_i \subseteq \mathcal{N}$ with $\mathcal{N}_i := \{n \mid \forall n \in \mathcal{N}_i : f(n) = c_i\}$. As we can see from the above definition there exist several sets of payload sizes $\mathcal{N}_i$ for which the resource consumption c_i in terms of minislots is constant, e.g., $c_2 = 3$ minislots will be consumed by any message having a payload size of n bytes where $n \in \mathcal{N}_2 = \{4, 6, 8, 10\}$. As a result, future messages that might be scheduled with priorities higher than m_i's will be considered by an estimated workload of c_x minislots. Hence, we account for their effective transmission time using $\overline{e}_x = (c_x - 1) \cdot t_{MS}$. Note that the choice of c_x captures the effective transmission time $\overline{e}_x$ for future messages that may have payload sizes $n \in \mathcal{N}_i$ for which $f(n) = c_x$. The relation in (3) allows for an expressive workload estimation as the system designer now does not need to know the

exact payload sizes of future messages. In fact, it is sufficient to specify a range of expected payload sizes which can be captured by a unique value of c_x, hence allows for an approximate payload estimation based on the designer's experience.

Sustainability test: In the following we present the schedulability condition for a sustainable schedule. Using (1) and (2) yields

$$d_i \geq R_i \cdot gdCycle + \max_{\forall k}\left(\sum_{j\in\mathcal{G}_k} \overline{e}_j + x_k \cdot \overline{e}_x\right) + e_i = \overline{D}_i \quad (4)$$

In other words, if a schedule Θ_i violates the deadline d_i in the presence of future messages then this schedule is not sustainable. Message sizes are estimated as to consume c_x minislots on an average. Second, we must consider the availability of minislots in the DYN segment which is captured by the FlexRay parameter $pLatestTx$ (see Section 2). Hence, we require that the maximum minislot counter value $\overline{\mu}_i$ (considering the workload of future messages c_x) must not exceed the value of $pLatestTx$:

$$\overline{\mu}_i = \max_{\forall k}\left(\sum_{j\in\mathcal{G}_k} c_j + x_k \cdot c_x\right) < pLatestTx. \quad (5)$$

If both (4) and (5) are fulfilled by any schedule Θ_i, such a schedule is referred to as sustainable. Using (4) and (5) we formulate the sustainability test as:

$$(d_i \geq \overline{D}_i) \wedge (\overline{\mu}_i < pLatestTx). \quad (6)$$

5. EXTENSIBILITY ANALYSIS

The sustainability test in the previous section does not characterize schedules according to the number of higher priority messages that may be accommodated in future. To account for this, we specify the notion of *extensibility* in what follows. Provided existing schedules are not changed at any time we quantify the following:

- the *quality rating* of slots, i.e., the ability of a slot S_i to provide real-time guarantees for future messages (e.g., slots with high priorities)
- the *grade of extensibility* to accommodate future messages in a slot S_i, i.e., the ability of a slot to provide versatile schedules Θ_i to accommodate future messages according to their real-time constraints
- the *extensibility index* which indicates a metric that quantifies extensibility according to the quality rating and the grade of extensibility of slots.

In the following, we elaborate the above-defined notions from the perspective of the FlexRay protocol-specific properties.

5.1 Quality rating of extensibility

In order to cope with a quantification of the quality rating of slots we introduce a quality rating function $P_1(S_i)$ which specifies a weight for every communication slot $S_i \in \mathcal{S}$. Effectively, $P_1(S_i)$ maps a weight to every slot according to its ability to provide real-time guarantees for future messages, e.g., according to slot priorities. We define a quality rating function $P_1(S_i)$ as follows:

$$P_1(S_i) = \begin{cases} 0, & \forall S_i \in \mathcal{R} \\ 1, & \forall S_i \in \mathcal{S}_{ST} \setminus \mathcal{R} \\ 1 - e^{-k\left(\frac{|S_i - S_{lD}|}{S_i - S_{fD}}\right)}, & \forall S_i \in \mathcal{S}_{DYN} \setminus \mathcal{R} \end{cases} \quad (7)$$

Equation (7) specifies different quality ratings for (i) *reserved* slots, (ii) *static* slots and (iii) *dynamic* slots which will be discussed in what follows.

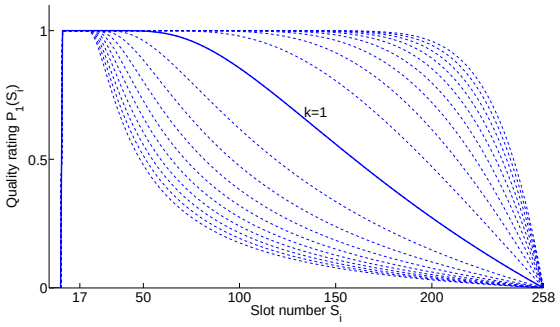

Figure 4: Quality rating of slots.

(i) Reserved slots: The first term in (7) accounts for reserved communication slots $S_i \in \mathcal{R}$ which are supposed to be used for system functions or specific protocols (see Section 3). For these slots the schedule assignment is often pre-defined according to specific rules or standards and hence we do not quantify their extensibility, i.e., $P_1(S_i)$ evaluates zero. In the example of Fig. 4, $\mathcal{R} = \{1, ..., 7\}$, i.e., the first 7 slots in the ST segment are reserved and hence cannot be assigned to application messages. Note, that slots can also be reserved in the DYN segment.

(ii) Static slots: The second term in (7) specifies the quality rating of static slots which are provided for scheduling application messages. All slots in the ST segment are of fixed, equal length and hence are of equal priority, i.e, messages cannot experience interference from any other messages transmitted in the ST segment. Consequently, we weight the static slots with a constant maximum rating factor $P_1(S_i) = 1,\ \forall S_i \in \mathcal{S}_{ST} \setminus \mathcal{R}$. For instance in Fig. 4, $P_1(S_i) = 1,\ \forall S_i \in \{8, ..., 17\}$.

(iii) Dynamic slots: The third term in (7) accounts for slots in the DYN segment that are not reserved, i.e., for $S_i \in \mathcal{S}_{DYN} \setminus \mathcal{R}$, where $\mathcal{S}_{DYN} = \{S_{fD}, ..., S_{lD}\}$. The event-triggered communication paradigm in the DYN segment implies that the quality rating function must change weights according to the slot priorities. However, the rate of change is not constant because (i) real-time guarantees can only be provided for messages assigned to *low* slot numbers (high priorities) where delays are bounded (see (1)) (ii) real-time guarantees cannot be provided for messages assigned to *high* slot numbers (low priorities) as messages can only be transmitted in a certain cycle if sufficiently many minislots are available according to *pLatestTx*. Therefore, we model the quality rating function for the DYN segment using a decreasing exponential function. According to (7), high priority slots S_i are quantified with a high quality rating, i.e., $P_1(S_i)$ evaluates towards 1 for $S_i \to S_{fD}$. Similarly, $P_1(S_i)$ evaluates towards zero for $S_i \to S_{lD}$. The system designer may configure the characteristics of $P_1(S_i)$ between S_{fD} and S_{lD} by adjusting the proportional factor $k > 0$ in (7). The choice of k depends on the FlexRay configuration and the design requirements. Fig. 4 illustrates $P_i(S_i)$ for different values of k and $\mathcal{S}_{DYN} = \{18, ..., 258\}$.

5.2 Grade of extensibility

So far we discussed how to quantify the quality rating of slots S_i in the ST and DYN segment. Next, we will discuss how to measure the *grade* of extensibility for a certain slot S_i. Let us consider the number of available schedules for a future message m_i that is scheduled in slot S_i. Within S_i, m_i can be assigned any of the admissible repetition rates $R_i \in \{1, 2, 4, 8, 16, 32, 64\}$ according to the protocol. For each of these repetition rates, there are R_i base cycle values available with $B_i < R_i$. For example, if $R_i = 2$, m_i might be assigned a base cycle $B_i \in \{0, 1\}$, if $R_i = 4$, then $B_i \in \{0, 1, 2, 3\}$, and so on. Thus, the total number of available

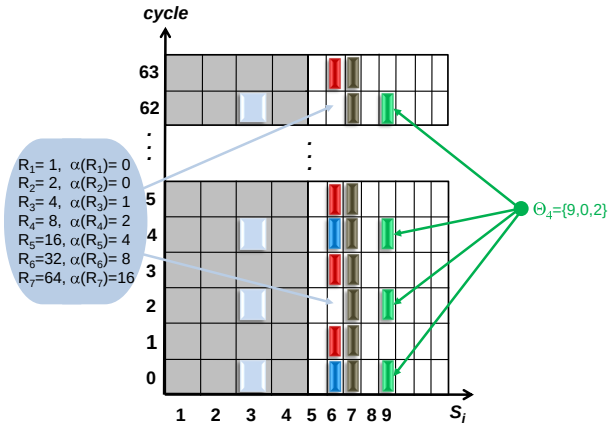

Figure 5: Quantification of extensible schedules.

choices to schedule m_i in slot S_i is computed as:

$$C(S_i) = \sum_{R_i} \alpha(R_i),\ \forall R_i \in \{1, 2, 4, 8, 16, 32, 64\} \quad (8)$$

where $\alpha(R_i)$ denotes the number of available base cycles $B_i < R_i$. As discussed above, $\alpha(R_i) = R_i$ if there is no other message scheduled in slot S_i. However, if existing schedules allocate cycles in slot S_i, and a message m_i is scheduled in the same slot using slot-multiplexing, then the number of available repetition rates R_i and base cycles B_i gets constrained because of the presence of the existing schedules. Let us consider the message m_4 with $\Theta_4 = \{9, 0, 2\}$ as depicted in Fig. 5 and study the influence of its schedule on the possible schedules for a future message m_i. For instance, $R_i = 1$, $B_i = 0$ is not available as the schedule $\Theta_i = \{9, 0, 1\}$ will lead to intersections with m_4's schedule in every even cycle as $\Gamma_4 \cap \Gamma_i = \{0, 2, 4, ..., 62\}$. Consequently, $\alpha(1) = 0$. Concerning the other repetition rates, only half of the choices are available for m_i, i.e., $\alpha(R_i) = \frac{R_i}{2}$ for $R_i \in \{2, 4, 8, 16, 32, 64\}$, as every even base cycle B_i is unavailable due to the presence of m_4's schedule and $C(S_i) = 0 + 1 + 2 + 4 + 8 + 16 + 32 = 63$. As the total number of available schedules in any empty slot $\hat{S}$ is computed as $C(\hat{S}) = \sum_{r=0}^{6} 2^r = 127$ we compute the grade of extensibility $P_2(S_i)$ of slot S_i as

$$P_2(S_i) = \frac{C(S_i)}{C(\hat{S})} \quad (9)$$

We now explain the grade of extensibility with another example. Consider the schedules $\Theta_1 = \{6, 1, 2\}$, $\Theta_2 = \{6, 0, 4\}$, $\Theta_3 = \{7, 0, 1\}$ and $\Theta_4 = \{9, 0, 2\}$ in the DYN segment as depicted in Fig. 5. Note from the figure, that every fourth cycle is available in slot 6. Using (9), we obtain $P_2(6) = \frac{0+0+1+2+4+8+16}{127} = 0.2441$, i.e., 24.41% of all schedules are available to schedule future messages in slot 6. Further, even though every fourth cycle is available, $P_2(6) \neq 0.25$ as future schedules where $R_i = \{1, 2\}$ are not available. Every cycle of slot 7 is assigned to Θ_3, hence $P_2(7) = 0$, and slot 9 is assigned to Θ_4 in every second cycle, thus $P_2(9) = 0.4961$. Note that this technique can be applied to the ST segment in a straightforward manner, e.g., for slot 3 with $\Theta_5 = \{3, 0, 2\}$ assigned to the even cycles (see Fig. 5), we obtain $P_2(3) = 0.4961$. In fact, this metric allows us to quantify the grade of extensibility to accommodate future messages in any slot S_i.

5.3 Extensibility index

Above we introduced two metrics related to extensibility. Using several extensibility metrics, the system designer needs to interpret different numbers, which makes a comparison between different schedules inherently difficult. Rather the designer needs to be able to make a meaningful statement on the extensibility properties of a FlexRay network in any design iteration I and to compare different schedules according to their extensibility, without perusing a list of numbers. Therefore, we introduce an *extensibility index* — a holistic quantification of extensibility that not only depends on the grade of extensibility, i.e., the number of available schedules in a slot, but also on the quality rating of a slot. As we will show in Section 6, this is an expressive metric which allows visual interpretation.

$$E(S_i) = P_1(S_i) \cdot P_2(S_i),\ \forall S_i \in \mathcal{S} \quad (10)$$

In particular, the extensibility index helps the system designer to observe the extensibility of the entire FlexRay schedule by quantifying available schedules for future messages and the quality rating of slots. Thus, resource bottlenecks in the system may be identified and suitable scheduling strategies may be pursued based on this.

6. EXPERIMENTAL RESULTS

In this section we illustrate the applicability of our proposed analysis techniques with industrial-size case studies.

Experimental setup: The FlexRay network parameters have been generated to be compliant with FlexRay using the SIMTOOLS configuration software [9]. We consider $gdCycle = 5ms$, $\mathcal{S}_{ST} = \{1, ..., 17\}$, $\mathcal{S}_{DYN} = \{18, ..., 258\}$, and $t_{MS} = 0.015ms$. Further, we assume reserved slots $S_i \in \mathcal{R} = \{1, ..., 7\}$ and a quality rating function of the form in (7). Fig. 4 in Section 5 illustrates the characteristics of quality ratings $P_1(S_i)$ over all slots $S_i \in \mathcal{S}$ with different values of k. In the following we use $P_1(S_i)$ with $k = 1$ as indicated by the continuous line in the figure. Message periods p_i, deadlines $d_i \leq p_i$, payload sizes $n_i \in \mathcal{N}$, and schedules $\Theta_i = \{S_i, B_i, R_i\}$, $\forall i \in \{1, ..., 100\}$ have been randomly generated in compliance with the FlexRay specification. We will consider an iterative design scenario with two consecutive iterations, $I \in \{1, 2\}$, where 100 messages are scheduled in the DYN segment at each iteration. For the generated schedules, we illustrate the proposed sustainability test in (6), and evaluate the extensibility index in (10) for both iterations. We illustrate (i) how our metrics allow easy visual interpretation and (ii) how extensibility and sustainability, taken together assist in understanding the nature of FlexRay schedules in the context of incremental scheduling.

Design iteration I=1: The grade of extensibility $P_2(S_i)$ is illustrated in Fig. 6. Note, that points in the figure where $P_2(S_i) = 0$ indicate slots where all cycles are allocated by schedules, e.g., $\Theta_1 = \{S_i, 0, 1\}$, or several slot-multiplexed schedules such as $\Theta_1 = \{S_i, 0, 2\}$ and $\Theta_2 = \{S_i, 1, 2\}$. This implies that no schedules are available for future messages with $S_i \in \Theta_i$. In contrast, points where $P_2(S_i) = 1$ denote empty slots where all feasible schedules are available for future messages, i.e., the grade of extensibility is maximum. This is especially prominent in the ST segment, i.e., $S_i \in \{1, ..., 17\}$, as no messages have been scheduled in static slots at any iteration I. Points in the range $0 < P_2(S_i) < 1$, indicate slots where one or more slot-multiplexed schedules partially allocate cycles in S_i and therefore constrain the number of available schedules for future messages according to (9). Fig. 7 depicts the extensibility index $E(S_i)$, $\forall S_i \in \mathcal{S}$ at design iteration $I = 1$. Recall that the extensibility index represents the grade of extensibility $P_2(S_i)$ weighted by the quality rating $P_1(S_i)$ for every $S_i \in \mathcal{S}$. Here, points along $P_1(S_i)$ indicate empty slots, i.e., full availability of schedules. As illustrated in the figure, $E(S_i)$ dramatically decreases for high slot numbers $S_i \rightarrow 258$ even if all schedules are available in a slot S_i. This accounts for the low-priority slots where future messages might experience high interference from other messages and hence, will less likely satisfy real-time guarantees. Thus, extensibility in those slots is weighted

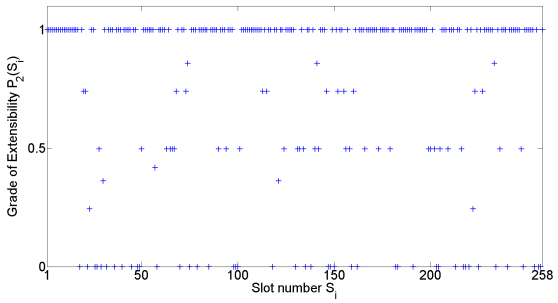

Figure 6: Grade of extensibility at I=1.

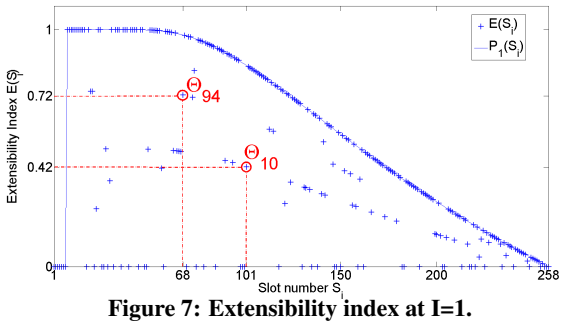

Figure 7: Extensibility index at I=1.

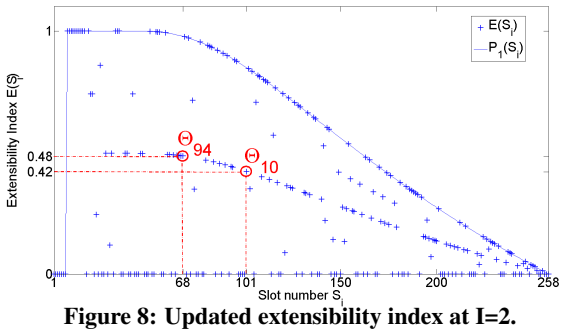

Figure 8: Updated extensibility index at I=2.

less according to the quality rating $P_1(S_i)$. On the other hand, Fig. 7 reflects a high grade of extensibility for low slot numbers indicating that these slots might easily accommodate high priority messages. Note that $E(S_i) = 0, \forall S_i \in \mathcal{R} = \{1, ..., 7\}$, because we do not consider reserved slots in our extensibility analysis according to $P_1(S_i)$. Further, $E(S_i) = 1, \forall S_i \in \mathcal{S} \setminus \mathcal{R}$ as no messages have been scheduled in the ST segment and hence $E(S_i) = P_2(S_i) = 1$. In essence, a visual reading of the extensibility index graph in Fig. 7 quickly reveals (i) the slots that are not extensible at all (points on the x-axis), (ii) slots that provide a high number of available schedules (points close to the curve $P_1(S_i)$ and (iii) slots that may accommodate high priority messages (points near the value of $E(S_i) = 1$).

Next, let us evaluate the sustainability test defined in (6). Towards this, consider future messages with a workload of $c_x = 4$ minislots which accounts for a payload range $n \in \{20, 22, 24, ..., 38\}$ in bytes following the discussion in Section 4. Consider for example message m_{10} with $\Theta_{10} = \{101, 1, 2\}$ which is marked in Fig. 7 at $E(101) = 0.42$. Currently, m_{10} easily meets its deadline constraints as deadline $d_{10} = 22ms > D_{10} = 11.305ms$ and the present minislot counter $\mu_{10} = 167 < pLatestTx = 238$. Hence, there is sufficient slack of $10.695ms$ and enough minislots are available to transmit m_{10} in its assigned cycles. However, considering future messages with $c_x = 4$ minislots resource consumption the sustainability test in (6) fails because the worst-case minislot counter evaluates $\overline{\mu}_{10} = 317$ which exeeds $pLatestTx$. Hence, Θ_{10} is not sustainable. Further, message m_{94} with $\Theta_{94} = \{68, 3, 4\}$ (see Fig. 7) also meets its deadline constraint at present because $d_{94} = 22ms > D_{94} = 21,56ms$ and the maximum minislot counter $\mu_{94} = 152 < pLatestTx = 238$. However, the sustainability test in (6) fails because the delay due to future messages is computed as $\overline{D}_{94} = 22,76ms > d_{94} = 22ms$ which might result in deadline violations at future design iterations $I > 1$.

Design iteration I=2: Fig. 8 illustrates the updated extensibility index at iteration $I = 2$ where 100 additional messages have been scheduled in the DYN segment. It can be noticed that the overall extensibility decreased as many points have dropped from the reference curve $P_1(S_i)$ compared to $I = 1$ (see Fig. 7). It is interesting to observe that high priority slots, that were not assigned previously, have now been assigned to the new messages, and this may be observed visually. For example in Fig. 8 at points $S_i \in \{18, ..., 50\}$, where $E(S_i) < P_1(S_i)$ compared to Fig. 7 where $E(S_i) = P_1(S_i)$. Compared to Fig. 7, many more points superimpose the x-axis implying that many slots are now completely allocated. Similarly, slots that have been partially allocated have now accumulated along a second curve segment around $0.5 \cdot E(S_i)$ as illustrated in Fig. 8, e.g., at slot $S_i = 68$ where m_{94} has been scheduled at the previous iteration (see Fig. 8). Now, $E(68)$ decreased from 0.72 to 0.48 between the two iterations, i.e., a new message has been scheduled at $I = 2$ using slot-multiplexing and hence $E(68)$ decreased accordingly. Finally, note that Θ_{94} and Θ_{10} were marked unsustainable in the previous iteration. It is interesting to observe that both of them now violate real-time requirements, due to the presence of interfering messages from iteration $I = 2$. In fact, Θ_{94} now violates its deadline constraints, i.e., $D_{94} > d_{94}$. For Θ_{10}, the current maximum minislot counter μ_{10} is computed as $\mu_{10} = 241 > pLatestTx$, i.e., m_{10} cannot be guaranteed to be transmitted in its assigned cycles.

7. CONCLUDING REMARKS

In this paper we specified a new notion of sustainability and extensibility in the context of the FlexRay protocol. Further, we proposed analysis techniques in order to identify sustainable and extensible schedules while introducing novel metrics reflecting the protocol details. The significance of the presented approach has been demonstrated with the help of experiments of industrial-size. For future work we envisage to incorporate the defined metrics to automatically synthesize sustainable and extensible schedules.

8. ACKNOWLEDGEMENT

The second author of this paper is an Alexander von Humboldt Research Fellow at TU Munich, Germany.

9. REFERENCES

[1] AUTOSAR. Specification of FlexRay Interface, Ver. 3.0.3. www.autosar.org.
[2] S. K. Baruah and A. Burns. Sustainable scheduling analysis. In *RTSS*, 2006.
[3] The FlexRay Communications System Specifications, Ver. 2.1. www.flexray.com.
[4] A. Ghosal, H. Zeng, M. D. Natale, and Y. Ben-Haim. Computing robustness of flexray schedules to uncertainties in design parameters. In *DATE*, 2010.
[5] M. Lukasiewycz, M. Glaß, P. Milbredt, and J. Teich. FlexRay Schedule Optimization of the Static Segment. In *CODES+ISSS*, 2009.
[6] P. Pop, P. Eles, T. Pop, and Z. Peng. An approach to incremental design of distributed embedded systems. In *DAC*, 2001.
[7] T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei. Timing analysis of the FlexRay communication protocol. *Real-Time Systems*, 39:205–235, 2008.
[8] E.G. Schmidt and K. Schmidt. Schedulability analysis and message schedule computation for the dynamic segment of flexray. In *VTC*, 2010.
[9] SIMTOOLS. www.simtools.at.
[10] H. Zeng, W. Zheng, A. Ghosal, P. Giusto, A. Sangiovanni-Vincentelli, and M. Di Natale. Scheduling and mapping in an incremental design methodology for distributed real-time embedded systems. In *DAC*, 2009.
[11] W. Zheng, J. Chong, C. Pinello, S. Kanajan, and A. Sangiovanni-Vincentelli. Extensible and scalable time triggered scheduling. In *Int'l Conference on Application of Concurrency to System Design*, 2005.