Using Schedule-Abstraction Graphs for the Analysis of CAN Message Response Times

Mitra Nasri, Arpan Gujarati, and Björn B. Brandenburg
Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern, Germany

***Abstract*—In a controller area network (CAN), electromagnetic interference (EMI) can result in message corruptions during transmission. The CAN protocol thus checks each message for corruption (using a checksum) and automatically schedules an erroneous message for retransmission. Retransmissions help tolerate EMI-induced errors with very high probability, but since EMI is stochastic in nature, they can affect the timing properties of the system in unpredictable ways. This work is about effectively quantifying the impact of retransmissions on the schedulability of real-time systems. Prior work focused on coarse-grained worst-case response-time analysis (RTA) of periodic or sporadic CAN message streams in the presence of retransmissions. For example, in case of periodic streams, prior analyses help upper-bound the maximum response time that any message belonging to a message stream might incur in the presence of a specific number of retransmissions. In this work, we present a fine-grained approach to analyze CAN message response times in the presence of retransmissions. The proposed analysis is based on the exploration of schedule-abstraction graphs, a novel abstraction for concisely capturing all possible schedules of CAN messages. Therefore, it enables upper-bounding the response times for each individual CAN message, and is not restricted to only periodic or sporadic message streams. We demonstrate the benefits of a message-specific analysis with a case study based on a simple mobile robot message set, and also discuss future opportunities enabled by such an analysis.**

I. Introduction

Embedded systems often need to operate in harsh environments, e.g., automotive embedded systems are surrounded by spark plugs and electric motors, industrial embedded systems in many cases are deployed in close vicinity to high-power machinery, and autonomous robots may need to operate in radiation-prone environments [1]. As a result, such embedded systems are susceptible to electromagnetic interference (EMI) and must be designed to withstand its effects [2].

In the context of real-time networked systems, EMI may result in frequent message corruptions on the network. To mitigate the effects of such corruptions, network stacks typically detect and retransmit the corrupted messages. For example, in a Controller Area Network (CAN), CAN controllers automatically queue messages for retransmission if any host signals a transmission error [3]. However, retransmissions may sometimes have a negative effect on the system reliability. Since EMI is stochastic in nature, retransmissions affect the timing behavior of the system in unpredictable ways, and may eventually compromise its functional safety due to deadline misses. This work is about effectively quantifying the effect of retransmissions on schedulability of CAN-based real-time systems.

Prior work in this regard [4, 5, 6, 7, 8, 9, 10] has focused on coarse-grained worst-case response-time analysis (RTA) of periodic and sporadic CAN message streams in the presence of retransmissions. For example, in case of periodic streams, prior analyses help upper-bound the maximum response time that any message belonging to a message stream might incur in the presence of a specific number of retransmissions.

The aforementioned analyses primarily rely on the classical technique of computing response times through fixed-point analysis of an iterated function [11]. As a result, although effective in detemining if a message set ever misses any deadline, they are often too coarse-grained for many use cases. For example, in case of weakly-hard systems [12] that can tolerate a few deadline violations (such as real-time control systems), a separate response-time analysis is required for each message in the message stream; as shown by Bernat et al. [12, 13], this is non-trivial with a fixed-point analysis. Even in the case of reliability analysis, using message stream-specific analyses to upper-bound the probability of a message transmission failure results in extremely pessimistic bounds on system realibiltiy [14].

Empirical approaches, e.g., [15, 16], provide detailed profiles of message-specific response times, but are not provably safe and thus problematic in the context of safety-critical systems.

In this work, we present a new approach based on schedule-abstraction graphs [17] to analyse both best-case and worst-case response times of individual messages belonging to each message stream in the workload. Schedule-abstraction graphs [17] are a recently proposed abstraction for concisely capturing all possible schedules of CAN messages. Therefore, they enable upper-bounding the response time of each individual CAN message with both release jitters and offsets, and are not restricted to only periodic or sporadic message streams.

To the best of our knowledge, this is the first instance of an analysis for CAN messages that explores all possible schedules while taking retransmissions into account.

The rest of paper is organized as follows. We start with an overview of prior work on schedule-abstraction graphs that is necessary to understand the proposed analysis (§II), and then explain in detail our retransmissions-aware analysis for CAN messages (§III). We then demonstrate the benefits of the proposed message-specific analysis with a case study based on a simple mobile robot message set and discuss some interesting open problems (§IV). Finally, we conclude with a brief discussion of future work (§V).

II. Schedule-Abstraction Graphs

The CAN protocol schedules message transmissions in the order of their fixed priorities [3]. Thus, schedulability analysis of CAN messages can be reduced to the problem of schedulability analysis of fixed-priority non-preemptive jobs on a uniprocessor platform.

In particular, we base our analysis on a recently proposed exact schedulabiltiy analysis of non-preemptive fixed-priority jobs [17], which uses *schedule-abstraction graphs* for efficiency. Traditionally, exact schedulability analyses have been developed for preemptive jobs using state exploration techniques based on model checking, timed automata, or linear-hybrid automata [18, 19, 20, 21]. These techniques do not apply to non-preemptive jobs and their sclability is limited. In contrast, the schedule-abstraction graph model, along with an effective merging strategy [17], allows for a more efficient representation and exploration of all possible uniprocessor schedules. In the following, we give a brief overview of this analysis.

Suppose a finite set of jobs $\mathcal{J}$ to be scheduled on a uniprocessor based on their priorities. Each job $J_i = ([r_i^{min}, r_i^{max}], [C_i^{min}, C_i^{max}], d_i, p_i)$ can be released at any time $r_i \in [r_i^{min}, r_i^{max}]$, has a transmission time of $C_j \in [C_i^{min}, C_i^{max}]$, an absolute deadline d_i, and a fixed priority p_i (a numerically lower value denotes a higher priority).

The schedule-abstraction graph for a job set $\mathcal{J}$ is a directed acyclic graph $G = (V, E)$ consisting of vertices V denoting the system states and edges E denoting job executions. Each edge (v_p, v_q, J_i) is directed from source vertex v_p to destination vertex v_q and is labeled with a job $J_i \in \mathcal{J}$, impliying that job J_i is executed between system states v_p and v_q, i.e., it is dispatched *next* after v_p or that it *succeeds* v_p. A system state $v_p = [A_p^{min}, A_p^{max}]$ represents an interval during which the processor is *possibly* available and at the end of which the processor is *certainly* available. The graph is rooted at the *initial* state $v_1 = [0, 0]$ denoting an idle processor. A possible schedule of any job set $\mathcal{J}^P \subseteq \mathcal{J}$ is modeled as a path P from the initial state v_1 to any state v_p such that the set of labels of edges on this path corresponds to $\mathcal{J}^P$.

Using the aforementioned graph-based abstraction for schedules, and given a particular scheduling policy, the BCRTs and the WCRTs of the jobs are determined through an iterative algorithm consisting of an expansion phase and a merging phase. In particular, during each iteration, a path P ending at vertex v_p is first *expanded* by deriving all jobs that can potentially be dispatched *next* after v_p through at least one valid execution scenario, i.e., through some valid assignment of release times and execition times. Afterwards, if any two paths P_1 and P_2 share the same set of jobs (i.e., $\mathcal{J}^{P_1} = \mathcal{J}^{P_2}$), their terminal vertices are compared. If the respective terminal vertices, say, v_p and v_q correspond to intersecting processor-availability intervals (i.e., $v_p \cap v_q \neq \emptyset$), then they are *merged* together to create a new state v_{pq} whose processor-availability interval is the union of the two (i.e., $v_{pq} = v_p \cup v_q$). In the merged state, paths P_1 and P_2 both terminate at vertex v_{pq}. The algorithm terminates when each path corresponds to $\mathcal{J}$.

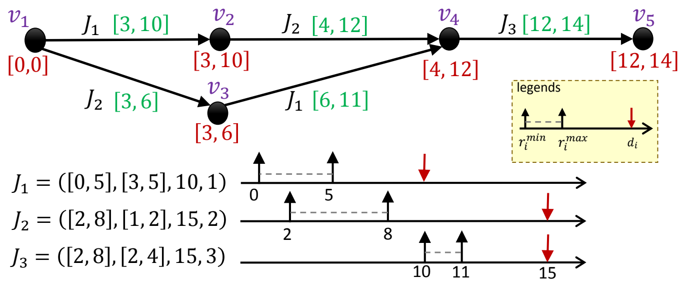

Fig. 1. A schedule-abstraction graph for three jobs $\mathcal{J} = \{J_1, J_2, J_3\}$. An interval after a job on an edge denotes the earliest and latest finish time of the job on that edge and an interval below a vertex represents when the processor becomes possibly and certainly ready in that state.

While constructing the graph, the algorithm stores the earliest and latest finish time of a job J_i for each edge $e = (v_p, v_q, J_i) \in P$, where J_i is dispatched next after v_p. Hence, upon the termination of the algorithm, the BCRT and the WCRT of job J_i is trivially obtained by finding its minimum and maximum finish times over all paths in G.

Example. Suppose that a given job set $\mathcal{J}$ consists of three jobs $J_1 = ([0, 5], [3, 5], 10, 1)$, $J_2 = ([2, 8], [1, 2], 15, 2)$, and $J_3 = ([2, 8], [2, 4], 15, 3)$. The schedule-abstraction graph for this job set is illustrated in Fig. 1. Due to the release jitter of J_1 and J_2, it is possible for either of them to be scheduled first, e.g., job J_2 is scheduled first if J_2 is released at time 3 and J_1 is released at time 4. Both possibilities are modeled in the graph by two outgoing edges labeled J_1 and J_2 incident to the initial state.

For each edge, an earliest and a latest finish time of the job that is scheduled first is calculated. These values are determined by taking into account the time at which a higher-priority job will certainly be released, the candidate job's release jitter interval and execution time variation, and the processor-availability interval of the source state of the edge. For example, when J_2 succeeds v_1 (and creates state v_3), its earliest start time is 2 and its latest start time is 4 since at time 5, a higher-priority job, i.e., J_1, will be certainly released and hence J_2 cannot be the highest-priority pending job from time 5 onward. Thus, the earliest and latest finish times of J_2 when it succeeds state v_1 are $2+1 = 3$ and $4+2 = 6$, respectively. On the other hand, if J_1 succeeds v_1, its earliest and latest finish times are 3 and 10 since its earliest and latest release times are 0 and 5, respectively, and its shortest and longest execution times are 3 and 5, respectively. When J_2 is dispatched after J_1, i.e., it succeeds state v_3, its earliest and latest start times will be 3 and 10, since the earliest and latest times, respectively, at which the processor becomes available after executing J_1 are 3 and 10. Using these values, the earliest and latest finish times of J_2 when it succeeds v_3 are 4 and 12, respectively.

Since both paths $\langle J_1, J_2 \rangle$ and $\langle J_2, J_1 \rangle$ share the same set of jobs and their intervals intersects, i.e., $[4, 12] \cap [6, 10] \neq \emptyset$, their terminal states are merged to obtain state v_4. Since J_3 is the only pending job at state v_4, the graph is trivially extended with edge (v_4, v_5, J_3). From this graph, the response time of J_1 along the path $\langle v_1, v_2, v_4, v_5 \rangle$ is $R_1 \in [3, 10]$, whereas its response time along the path $\langle v_1, v_3, v_4, v_5 \rangle$ is $R_1 \in [6, 11]$. Thus, J_1's BCRT and WCRT is $\min(3, 6) = 3$ and $\max(10, 11) = 11$, respectively. Jobs J_2 and J_3's response-time bounds can be computed similarly.

A detailed proof of correctness of the analysis can be found in [17]. Further, the paper provides a thorough discussion on how to use the proposed schedulability analysis for periodic tasks with constrained deadlines and/or release offsets.

III. Retransmissions-Aware Analysis

Using the schedule-abstraction graph-based analysis discussed in §II as a black box, we next propose a technique to analyze CAN message response times while accounting for fault-induced retransmissions.

Consider a finite set of CAN messages $\mathcal{M}$ where each message $M_i \in \mathcal{M}$ can be released at any time $r_i \in [r_i^{min}, r_i^{max}]$, has a transmission time of $C_j \in [C_i^{min}, C_i^{max}]$, an absolute deadline d_i, and a fixed priority $p_i \geq 1$. Release time variation for CAN messages is common due to scheduling delays, queuing delays, buffering, etc. Transmission time variation occurs due to changes in data values or bit-stuffing[1] and is typically small. For an 8-byte data frame, for example, the frame size can vary between 108 and 126 bits.

Without retransmissions, an exact value of the WCRTs can be trivially estimated using schedule-abstraction graphs since $\mathcal{J} \triangleq \mathcal{M}$. With retransmissions though, deriving an exact WCRT or even an upper bound on the exact WCRT of a message is challenging since errors that cause retransmissions happen in a non-deterministic way. In the following, we propose an analysis to derive an upper bound on the exact WCRT of any message $M_i \in \mathcal{M}$ given that up to f retransmissions may happen during the time the message set is transmitted over the network. We denote the exact value of this retransmissions-aware WCRT of each message M_i as $R_i(f)$.[2]

We assume that each host transmitting messages on CAN has enough buffer to store all pending messages, including the messages that are scheduled for retransmission.

Analysis. A schedule of $\mathcal{M}$ with f retransmissions means that a total of $n + f$ messages are actually transmitted over CAN, including n successful transmissions and f erroneous transmissions. Thus, to use schedule-abstraction graphs, we consider a revised message set $\mathcal{M}' = \mathcal{M} \cup \mathcal{M}^f$ where $\mathcal{M}^f = \{M_{n+1}, M_{n+2}, \ldots, M_{n+f}\}$. Namely, each message in $\mathcal{M}$ represents a *successful transmission* and each message in $\mathcal{M}^f$ represents an erroneous transmission over the network.

Messages in $\mathcal{M}^f$ are defined as follows. Since messages can be corrupted at any time in a non-deterministic way, we model the release of each message in $\mathcal{M}^f$ using the release jitter interval $[r^{min}, d^{max}]$, where $r^{min} \triangleq \min\{r_i^{min} \mid M_i \in \mathcal{M}\}$ denotes the earliest possible release event and $d^{min} \triangleq \max\{d_i \mid M_i \in \mathcal{M}\}$ denotes the latest possible deadline in message set $\mathcal{M}$. Since any message in $\mathcal{M}$ can be corrupted, the transmission times of messages in $\mathcal{M}^f$ are modeled as ranging from $C^{min} \triangleq \min\{C_i^{min} \mid M_i \in \mathcal{M}\} + \epsilon$ up to $C^{max} \triangleq \max\{C_i^{max} \mid M_i \in \mathcal{M}\} + \epsilon$. The error overhead ϵ denotes the time corresponding to the transmission of an error frame on the network, which happens immediately after the transmission of an erroneous message. We also set the priority of each erroneous message to the highest priority, i.e., zero, to model the corruption of the highest-priority message in the worst case (recall that $\forall M_i \in \mathcal{M}, p_i \geq 1$). The deadline of each erroneous message is irrelevant and set to ∞.

To summarize, the k^{th} erroneous message M_{n+k} is defined as $M_{n+k} \triangleq \big([r^{min}, d^{max}], [C^{min}, C^{max}], \infty, 0\big)$. Thanks to this definition, an erroneous message **(i)** can be released *anytime* in the window of interest, i.e., $[r^{min}, d^{max}]$, and **(ii)** has a priority higher than any message in $\mathcal{M}$ and hence can be transmitted before any message in $\mathcal{M}$. As a result, the transmission of erroneous messages can affect the WCRT of *any successfully transmitted message*. Invoking the schedule-abstraction graph-based analysis [17] with message set $\mathcal{M}'$ thus yields a safe upper bound on the WCRT of messages in $\mathcal{M}$, given that they may be affected by up to f retransmissions. Note that we are not interested in the WCRT of the erroneous messages in $\mathcal{M}^f$, but only on their impact on the response times of correctly transmitted messages.

We next explain the schedule-abstraction graph generated for a set of CAN messages, including the erroneous messages that we model for a black-box analysis, using a simple example.

Example. Suppose that message set $\mathcal{M}$ consists of two messages $M_1 = ([0, 5], [3, 5], 14, 1)$ and $M_2 = ([6, 6], [1, 2], 30, 2)$. The schedule-abstraction graph for $f = 2$ is illustrated in Fig. 2. M_3 and M_4 denote the two erroneous messages in this case. The analysis keeps track of the largest observed value of WCRT of each correct message in all paths. In this example, $R_1(2) = 15$ and $R_2(2) = 22$, which means that M_1 misses its deadline at time 14. This happens when M_1 is transmitted after two erroneous trials, shown by the two paths $\langle M_3, M_4, M_1, M_2 \rangle$ and $\langle M_4, M_3, M_1, M_2 \rangle$. A scenario in which neither M_1 nor M_2 is transmitted erroneously is represented by the path $\langle M_1, M_2, M_3, M_4 \rangle$. In this path, the response times of M_1 and M_2 are not affected by any retransmission. Note that message M_1 is always certainly released before message M_2 is released. Since M_1 has a higher priority than M_2, message M_2 can never be transmitted as long as M_1 has not been transmitted.

IV. Case Study

In this section, we demonstrate the benefits of the proposed fine-grained message-specific analysis with a case study based on a simple mobile robot message set. In particular, we show that the proposed analysis helps to **(i)** account for weakly-hard timing constraints, and to **(ii)** assign message offsets such that

[1]CAN controller inserts a bit of opposite polarity after five consecutive bits of the same polarity while transmitting any data on the network. This practice is called *bit stuffing*, and is necessary due to the non-return to zero (NRZ) coding used with CAN. The stuffed data frames are destuffed by the receiver.

[2]Even though f is non-deterministic in practice, it could be estimated through an empirical analysis of EMI on the CAN bus under different types of operational environments.

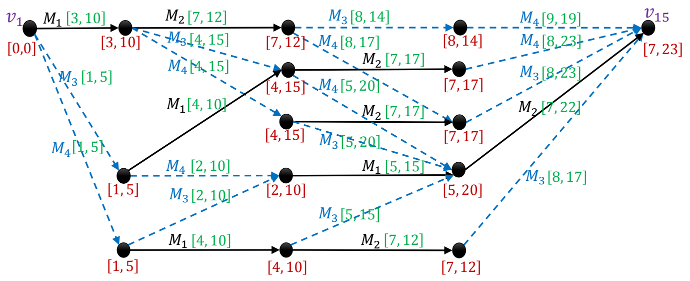

Fig. 2. A schedule-abstraction graph for $f = 2$ and $\mathcal{M} = \{M_1, M_2\}$, where $M_1 = ([0, 5], [3, 5], 14, 1)$ and $M_2 = ([6, 6], [1, 2], 30, 2)$. Dashed lines denote erroneous transmissions.

TABLE I
MOBILE ROBOT MESSAGE SET FROM [6]

Benchmark	id	priority	length(μs)	period(μs)	deadline(μs)
MotorCtrl	$\overline{M}_1$	1	288	2000	2000
Wheel1	$\overline{M}_2$	2	328	4000	4000
Wheel2	$\overline{M}_3$	3	328	4000	4000
RadioIn	$\overline{M}_4$	4	528	8000	8000
Proximity	$\overline{M}_5$	5	248	12000	12000
Logging	$\overline{M}_6$	6	528	240000	12000

the overall distribution of the WCRTs is improved. Later, we discuss some interesting open problems for future work.

The employed mobile robot benchmark [6] is designed for a CAN bus with a bit-transmission rate of 256 Kbps. It consists of six periodic message streams, denoted by $\overline{M} = \{\overline{M}_1, \overline{M}_2, \ldots, \overline{M}_6\}$ as listed in Table I.

We convert these periodic message streams to a finite set of messages (as required by the analysis in §III) as follows. Let T_i denote the periods of the respective message streams, and $H = lcm(T_1, T_2, \ldots, T_6)$ denote their hyperperiod, where lcm denotes the least common multiple. Starting with $\mathcal{M} = \emptyset$, for each message that belongs to message stream $\overline{M}_i$, say, the j^{th} message, and that is released during the hyperperiod, we add a new message $M_{i,j}$ to $\mathcal{M}$. Each message $M_{i,j}$ has a release interval $[(j-1)T_i, (j-1)T_i + \delta]$, where δ denotes the maximum release jitter of $\overline{M}_i$. The best-case transmission time for each message $M_{i,j}$ is $C_{i,j}^{min} = 72$, corresponding to the transmission time of the minimum-sized (one byte) packet, and the worst-case transmission time for each message $M_{i,j}$ is assigned as per the "length" column in Table I.

For our experiments, we considered $\delta = 10\mu$s and $f = 1$, i.e., each message is affected by up to one retransmission in a hyperperiod, or in other words, at most one message is erroneously transmitted. In practice, δ is upper-bounded through a careful analysis of the system processes that generate the messages to be transmitted, and f can be estimated with high confidence based on the peak rate of EMI, which in turn is known from empirical measurements and/or environmental modeling. All experimental results are illustrated in Fig. 3.

As mentioned in §I, the proposed analysis allows a detailed study of the WCRTs of each message stream in the workload. This enables the analysis of other higher-level properties and guarantees, for instance such as whether any message stream incurs more than three consecutive deadline misses (i.e., weakly-hard constraints [12]), or opportunities for a reduction in response-time jitter. We next discuss some relevant examples that demonstrate the use of the proposed analysis in these ways in the context of the case study.

Understanding the response-time distributions. Figs. 3(a)-3(e) show the WCRTs of the first 15 messages belonging to message streams $\overline{M}_1$-$\overline{M}_5$, respectively. For message stream $\overline{M}_6$, only a single message is transmitted during the hyperperiod with a WCRT of $R_{6,1}(1) = 3187\mu$s. We observe that the WCRTs for each message stream follow a repeating pattern depending on the relation between the different message periods. For example, as shown in Fig. 3(g), messages belonging to message stream $\overline{M}_5$ have three different types of WCRTs, and from the second message onward, the WCRTs alternate. In this case, $M_{5,1}$ has the largest WCRT compared to subsequent messages of $\overline{M}_5$ because its arrival time coincides with that of $M_{1,1}, M_{2,1}, \ldots, M_{6,1}$. Since the release jitter $\delta = 10\mu$s for each message, any of these messages can be scheduled before $M_{5,1}$. However, $M_{5,2}$ arrives at time 12000μs, when $M_{4,1}$, $M_{4,2}$, and $M_{6,1}$ are no longer pending and hence their transmission does not affect the WCRT of $M_{5,2}$. Consequently, after $M_{5,1}$'s transmission, the WCRT of each $M_{5,j}$ varies alternately depending on whether a message instance of $\overline{M}_4$ or $\overline{M}_6$ arrives at the same time as $M_{5,j}$. It is worth noting that in the presence of jitter, both lower- and higher-priority messages may interfere with a message instance.

Verifying weakly-hard constraints. Weakly-hard constraints are usually represented in the form of (m, k) constraints, i.e., *at least m message instances must arrive before their deadline in any consecutive sequence of k message instances* [12]. Hence, a fine-grained knowledge about the WCRT of message instances is required to evaluate whether or not a message stream conforms a weakly-hard timing constraint. Since our analysis derives the WCRT of each message during a hyperperiod, verifying an (m, k) constraint for a message stream $\overline{M}_i$ is equivalent to counting the number of deadline misses in each window of k messages and ensuring that at least m messages within that window are transmitted before their deadlines.

Reducing WCRTs using offset assignment. Assigning offsets allows a system designer to avoid creating a large amount of interference (i.e., long busy windows) and hence improves schedulability [22]. This, however, requires having a fine-grained WCRT analysis per message instance in order to understand which individual messages suffer from large interference and how message streams should be aligned using initial offsets so that their response times are reduced.

For example, the WCRT of messages in the mobile robot message set (Table I) can be reduced through offset assignment as follows. We use $O_1 = 0\mu$s, $O_2 = 0\mu$s, $O_3 = 2000\mu$s, $O_4 = 0\mu$s, $O_5 = 2000\mu$s, and $O_6 = 4700\mu$s as the respective offsets. Since every period in $\overline{M}$ is an integer multiple of $T_1 = 2000\mu$s, by aligning other message streams with either the odd or the even message instances of $\overline{M}_1$, the interference among message

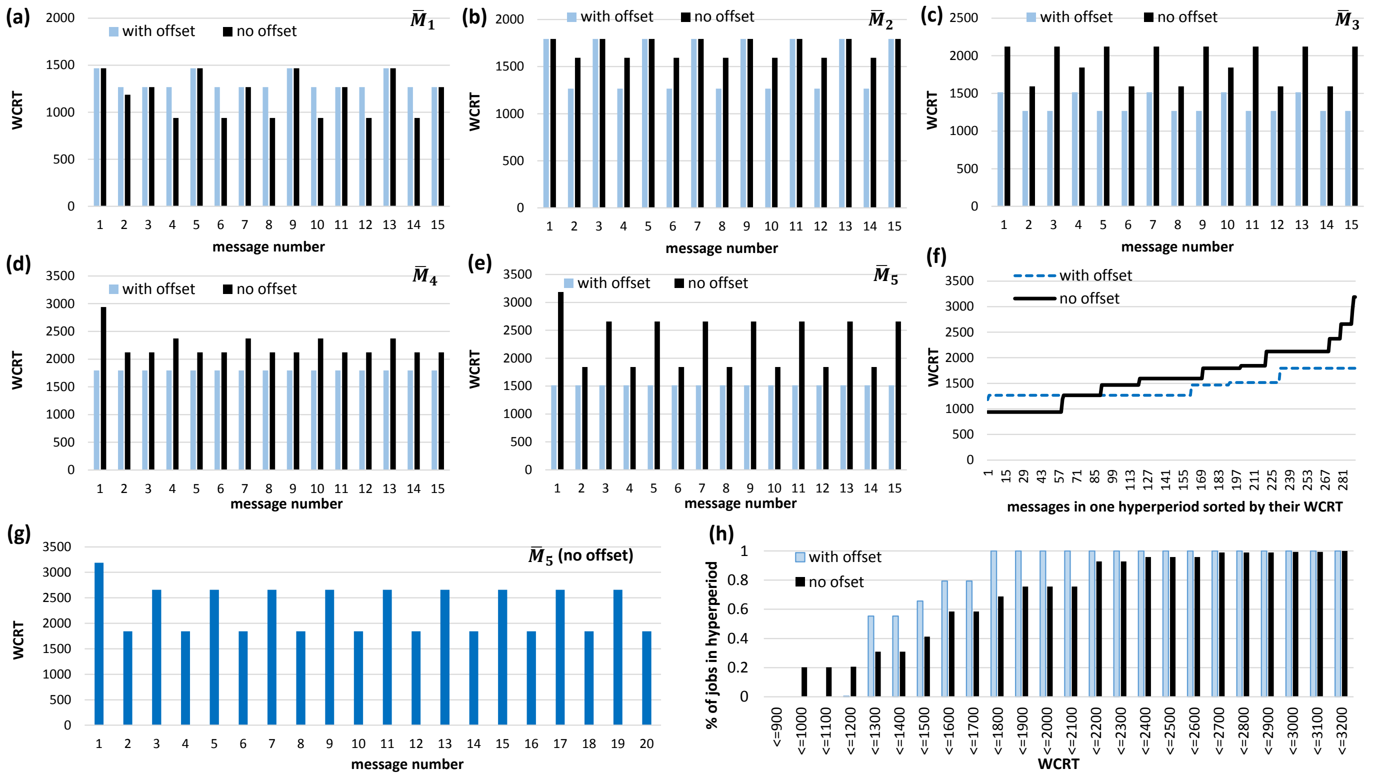

Fig. 3. WCRT of jobs for the benchmark message set in Table I: Experimental results for task sets with zero, small, and large jitter. **(a, b, c, d, e)** WCRT of the first 15 jobs of messages $\overline{M}_1$ to $\overline{M}_5$, **(f)** WCRT of each individual message in one hyperperiod sorted in a non-decreasing order, **(g)** WCRT of all jobs of message $\overline{M}_5$ in a synchronous-release scenario, and **(h)** cumulative distribution function of message instances for each WCRT scale from 900 to 3200.

streams can be reduced (see Fig. 3). In particular, using the chosen offsets, the WCRT of message streams $\overline{M}_i, i \geq 2$ is reduced efficiently while only a few instances of $\overline{M}_1$ experience an increase in their WCRT. Note that assigning large offsets to message streams may result in carry-in workload for the next hyperperiod which, in turn, increases the length of the observation window that must be analyzed [17, 23]. We are hence interested in offset assignments that do not push extra workload to the next hyperperiod.

Fig. 3(f) shows the response time of all individual messages over a hyperperiod. To make the trends clear and visible, we sorted the WCRTs in ascending order. As it can be seen, our offset assignment resulted in a more balanced WCRT distribution over the hyperperiod. In particular, it reduced the tail of the distribution of WCRTs (see Fig. 3(h)).

Discussion. As shown before, a fine-grained response-time analysis, which takes into consideration the WCRT of individual messages rather than a message stream, allows verifying more general forms of timing constraints, e.g., weakly-hard constraints. It also provides directives on how to find offsets such that a more balanced distribution of WCRT is attained for a message stream. In the following, we discuss some open problems that focus on a combination of practical constraints in designing robust and reliable CAN-based systems.

CAN controllers usually have a buffer to store pending messages, including messages that are released but not yet send and messages that have been transmitted erroneously and must be retransmitted. The WCRT of a message instance is an indicator of the lifetime of that individual message in the CAN controller buffer of the node it belongs to. Thus, taking into account the arrival time and WCRT of message instances, it is possible to derive the buffer size of the CAN controller on each node. This leads to the first interesting open problem.

Open Problem 1. *Given a schedulable message set $\mathcal{M}$ and the number of retransmissions f affecting each message, derive an upper bound on the required buffer size of each CAN node.*

Open Problem 2. *Given a fixed buffer size B for each CAN controller, a message set $\mathcal{M}$, and number of retransmissions f, derive the WCRT of each message while accounting for messages dropped due to buffer overflows.*

Open Problem 2 can be extended to systems with weakly-hard constraints in order to evaluate the schedulability of a message set using an (m, k) constraint.

Another direction is to devise a more accurate, ideally exact, response-time analysis for message sets with retransmission, where instead of an upper bound on the WCRT, an exact WCRT of each message is obtained. In our current approach, each erroneous transmission is modeled as a non-deterministic event (erroneous message) which can happen before any message and its worst-case transmission time is as large as the largest message in $\mathcal{M}$. This approach, however, contains two main sources of pessimism: **(i)** it pessimistically increases the response time of (high-priority) messages with short transmission time since the transmission time of the erroneous message is set to be as large as C^{max}, which might be determined by a low-priority message, and **(ii)** it includes scenarios where more than one retransmission of a lower-priority message can possibly happen before a pending higher-priority message is transmitted. The latter situation happens because we assign the highest priority to the erroneous messages. However, in reality, a retransmission can only happen in the order of priority of pending messages. Namely, a higher-priority message can be blocked by at most one retransmission of a lower-priority message. To remove this pessimism, failed transmissions must be incorporated into the generation of the schedule-abstraction graph such that a failed transmission inherits the properties of the last message dispatched on a path. This requires defining new rules for *expanding* and *merging* paths in the graph and hence requires its own proof of correctness afterwards, which we leave to future work.

V. Conclusion

The paper provides a sufficient schedulability analysis for a set of messages transferred over a CAN bus in the presence of message retransmission due to transient errors caused by electromagnetic interference. The analysis derives the worst-case response time (WCRT) of each individual message as a function of the maximum number of bit-flips (errors) that can happen within the given window of time. The paper explains how to use a state-of-the-art exact schedulability analysis of a set of non-preemptive jobs upon a uniprocessor platform in order to obtain an upper-bound on the WCRT of the messages in the presence of transient errors. Since the analysis is message-specific, it opens up an array of opportunities as discussed in §IV. In particular, it would be interesting to derive an exact retransmission-aware response-time analysis by directly modifying the way the schedule-abstraction graph is explored.

Acknowledgements.

The first author is supported by a post-doctoral fellowship awarded by the Alexander von Humboldt Foundation.

References

[1] C. Johnson, *Failure in Safety Critical Systems: A Handbook of Incident and Accident Reporting Glasgow University Press*. Glasgow, 2003.

[2] J. Noto, G. Fenical, and C. Tong, “Automotive EMI shielding–controlling automotive electronic emissions and susceptibility with proper EMI suppression methods,” *Laird Technologies*, 2010.

[3] M. Di Natale, H. Zeng, P. Giusto, and A. Ghosal, *Understanding and Using the Controller Area Network Communication Protocol*. Springer New York, 2012.

[4] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, “Controller area network (CAN) schedulability analysis: Refuted, revisited and revised,” *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.

[5] S. M. T. de Carvalho and G. L. Campos, “Worst case response time approach evaluation for computing CAN messages response time in an automotive network,” in *Brazilian Power Electronics Conference*, 2017, pp. 1–6.

[6] I. Broster, A. Burns, and G. Rodríguez-Navas, “Timing analysis of real-time communication under electromagnetic interference,” *Real-Time Systems*, vol. 30, no. 1-2, pp. 55–81, 2005.

[7] S. Punnekkat, H. Hansson, and C. Norstrom, “Response time analysis under errors for CAN,” in *IEEE Real-Time Technology and Applications Symposium (RTAS)*, 2000, pp. 258–265.

[8] P. M. Yomsi, D. Bertrand, N. Navet, and R. I. Davis, “Controller area network (CAN): Response time analysis with offsets,” in *IEEE International Workshop on Factory Communication Systems (WFCS)*, 2012, pp. 43–52.

[9] Y. Dong, H. Ma, H. Xu, and K. Wen, “Response time analysis of mixed scheduling over CAN,” in *International Conference on Future Computer and Communication (ICFCC)*, 2010, pp. 494–498.

[10] L. Pinho, F. Vasques, and E. Tovar, “Integrating inaccessibility in response time analysis of CAN networks,” in *IEEE International Workshop on Factory Communication Systems (WFCS)*, 2000, pp. 77–84.

[11] M. Joseph, “Finding response times in a real-time system,” *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.

[12] G. Bernat, A. Burns, and A. Liamosi, “Weakly hard real-time systems,” *IEEE Transactions on Computers*, vol. 50, no. 4, pp. 308–321, 2001.

[13] Guillem Bernat Nicolau, “Specification and analysis of weakly hard real-time systems,” PhD thesis, Universitat de les Illes Balears, Spain, 1998.

[14] A. Gujarati, M. Nasri, and B. B. Brandenburg, “Quantifying the resiliency of fail-operational real-time networked control systems,” in *Euromicro Conference on Real-Time Systems (ECRTS)*, 2018.

[15] J. Diaz, D. Garcia, Kanghee Kim, Chang-Gun Lee, L. Lo Bello, J. Lopez, Sang Lyul Min, and O. Mirabella, “Stochastic analysis of periodic real-time systems,” in *IEEE Real-Time Systems Symposium (RTSS)*, 2002, pp. 289–300.

[16] Y. Lu, T. Nolte, J. Kraft, and C. Norstrom, “A statistical approach to response-time analysis of complex embedded real-time systems,” in *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2010, pp. 153–160.

[17] M. Nasri and B. B. Brandenburg, “An exact and sustainable analysis of non-preemptive scheduling,” in *IEEE Real-Time Systems Symposium (RTSS)*, 2017, pp. 1–12.

[18] T. P. Baker and M. Cirinei, “Brute-force determination of multiprocessor schedulability for sets of sporadic hard-deadline tasks,” in *International Conference on Principles of Distributed Systems (OPODIS)*. Springer, 2007, pp. 62–75.

[19] A. Burmyakov, E. Bini, and E. Tovar, “An exact schedulability test for global FP using state space pruning,” in *International Conference on Real-Time Networks and Systems (RTNS)*, 2015.

[20] N. Guan, Z. Gu, Q. Deng, S. Gao, and G. Yu, “Exact schedulability analysis for static-priority global multiprocessor scheduling using model-checking,” in *Software Technologies for Embedded and Ubiquitous Systems (SEUS)*, 2007, pp. 263–272.

[21] Y. Sun and G. Lipari, “A pre-order relation for exact schedulability test of sporadic tasks on multiprocessor global fixed-priority scheduling,” *Real-Time Systems*, vol. 52, no. 3, pp. 323–355, 2016.

[22] K. W. Tindell, A. Burns, and A. J. Wellings, “An extendible approach for analyzing fixed priority hard real-time tasks,” *Real-Time Systems*, vol. 6, no. 2, pp. 133–151, 1994.

[23] J. Goossens, E. Grolleau, and L. Cucu-Grosjean, “Periodicity of real-time schedules for dependent periodic tasks on identical multiprocessor platforms,” *Real-Time Systems*, vol. 52, no. 6, pp. 808–832, 2016.