Computation of the Precise Worst-Case Response Time of FlexRay Dynamic Messages
Lucien Ouedraogo, Member, IEEE  Ratnesh Kumar, Fellow, IEEE

Abstract—FlexRay is a communication bus (and associated protocol) that supports transmission of time-triggered and event-triggered frames. A method for determining the worst-case response-time of FlexRay frames is proposed by Pop et al. in 2008, and is formulated as iterative sequence of Integer Linear Programming (ILP) problems. As we show, the method of Pop et al. is conservative (overestimates the response time). We propose a new ILP formulation that computes a precise value of the worst-case response time of FlexRay frames transmitted in the dynamic segment. Furthermore, our approach is non-iterative as it requires the solving of a single ILP for computing respectively the delay of full bus cycles and the delay of a partial (last) bus cycle. The proposed solution is also validated by applying it to a SAE benchmark and can be used for formally guaranteeing that no message will miss its deadline during system operation.

Note to Practitioners: To develop FlexRay-based distributed systems, Engineers need tools for schedulability analysis to be able to assert that a given network configuration parameters, e.g. frames priority and periodicity, meet the timing requirements of their distributed application (no deadline misses). Achieving this task includes the ability to compute the end-to-end delay of every frame. Practitioners can find in this paper an efficient method for precisely computing the worst-case response time, which is a significant part of the en-to-end-delay, of FlexRay frames scheduled in the dynamic segment. The solution can be easily implemented in FlexRay networks schedulability analysis tools.

Index Terms—FlexRay, In-vehicle network, Timing analysis, Real-time embedded systems, Integer linear programming.

I. INTRODUCTION

The FlexRay protocol [2]–[4], developed by a large consortium of car manufacturers, is a communication protocol intended to be used for high speed, deterministic and fault-tolerant communication. Compared to pure TDMA-based protocols such as the Time-Triggered Protocol (TTP) [5] that are suitable for transmission of time-triggered (TT) messages or CSMA-based protocols such as CAN that are suitable for transmission of event-triggered (ET) messages, FlexRay combines the advantages of these two approaches by allowing transmission of TT messages and ET messages in dedicated static (ST) and dynamic (DYN) segments respectively that improve the bus utilization. In order to be used for applications with hard real-time requirements, techniques for determining the response times of messages transmitted on a FlexRay communication bus are needed. This paper provides an efficient method for precisely computing the worst-case response time of FlexRay messages given timing characteristic (periodicity, jitters, deadlines) of the messages and the mapping of the messages onto the different FlexRay slots along with the message priorities. The authors of [6] provide an analysis of Flexible FTDMA (FTDMA) that finds the maximum frame identifier for messages so that the delays are reduced. However, this method cannot be applied directly to FlexRay because of the influence of the ST segment. Probabilistic approaches have also been used for timing analysis of FlexRay [7]–[9], which determine the average values instead of the precise values of the response time of DYN messages, and thus they are not useful for time critical systems. The authors of [10], [11] propose an analytical framework based on real-time calculus for performances analysis of FlexRay, that allows computation of upper bounds of the response time of messages.

The authors of [12] show the computation of worst-case response time to be similar to a bin covering problem for which an optimal solution can be computed using an ILP formulation (NP-hard problem). Their ILP formulation is conservative and the worst-case response time may be smaller. This is because the ILP is executed iteratively in order to account for all possible occurrences of higher priority DYN messages in a given time interval, but their minimum interarrival times have not been enforced adequately.

An attempt to potentially circumvent the limitation of [12] is presented in [13] by adding a constraint on inter departure times (as opposed to inter arrival times). It turns out that this constraint is stronger than what is physically possible, and as a result [13] yields an optimistic result (a lower bound) for the response time.

In this paper, we propose a new ILP formulation that accurately takes into account the minimum interarrival time of messages and thereby allows to precisely compute the value of the worst-case response time of DYN messages. Moreover our ILP formulation is non-iterative and it allows to compute the number of filled cycles in the worst-case response time in only one iteration, contrary to the formulation of [12] in which the ILP program is solved iteratively until all the possible instances of higher priority DYN messages are taken into account. Thus we are able to provide a more efficient solution (requiring a single execution of ILP as opposed to multiple...
executions) that is precise (as opposed to conservative). The implementation of a non-iterative ILP solution can be realized with less underlying labor due to the availability of numerous high quality ILP solvers.

As a proof of concept, we implemented our formulation as well as the formulation of [12] for a SAE benchmark consisting of 31 sporadic messages, and report the improvement in precision as well as efficiency. As periodic messages are generally scheduled in the ST segment and only aperiodic messages are scheduled in the DYN segment, the number of DYN messages for a specific application may not be too large to be handled by an ILP solver in a reasonable time. For example, a FlexRay network-based chassis control system realized with only 21 ST messages and 8 DYN messages is studied in [14].

The paper is organized as follows. In Section II, we describe the FlexRay Protocol. In Section III, we present in detail the analysis of FlexRay messages response time. Section IV contains our experimental results and Section V the concluding remarks.

II. THE FLEXRAY PROTOCOL

A. Overview of the FlexRay bus and Protocol

A FlexRay communication system consists of nodes communicating through one or two channels. A node consists of a host computer (CPU) running software processes (collection and processing of data) and a Communication Controller (CC) which implements the FlexRay protocol services. The CPU and the CC communicate through a Controller-Host Interface (CHI). The communication in a FlexRay system takes place in periodic cycles following a timing hierarchy.

At the highest level, the application cycle consists of 64 communication cycles (or bus cycles). The second level is the communication cycle level, where a cycle is itself divided into four possible segments: a mandatory static (ST) segment, an optional dynamic (DYN) segment, an optional symbol window (SW) and a mandatory network idle time (NIT).

The ST segment uses a static Time Division Multiple Access (TDMA) scheme for the media access control. Therefore, the ST segment is divided into equal length slots, and the transmission of a given frame in a slot can occupy either a part or the entire slot. On the other hand, the DYN segment uses a flexible TDMA, a dynamic mini-slotted based scheme: The DYN segment is divided into equal length minislots of smaller durations compared to the slots of the ST segment. The duration of a DYN slot depends on whether or not communication takes place. The DYN slot will consist of only one minislot if no communication takes place, and otherwise the DYN slot will consist of the number of minislots necessary for transmitting that frame. Moreover, a DYN frame is transmitted in a cycle if its assigned dynamic slot appears before the specified latest transmission time of its node. Arbitrations within the static segment and the dynamic segment are based on the unique assignment of frame identifiers to the nodes and a counting scheme for the slots and minislots. Each node is allowed to transmit frames in its assigned slots or minislots. To ensures this arbitration, the nodes are synchronized to the same clock. The SW is used for transmission of predetermined symbol(s) for the protocol operation control. During the NIT, no transmission occurs on the bus, and this delay is used by the nodes for calculating and applying the clock correction terms.

At the lower levels, the respective lengths of every minislot, slot and segment of the communication cycle consists of a fixed integer number of macroticks, where each macrotick consists in itself an integer number of microticks. Microticks are node-dependent time unit based on the ticks of the CC clock oscillator, while macroticks have a fixed duration over synchronized nodes.

Figure 1(a) presents an empty FlexRay communication cycle of a FlexRay network where the ST segment is divided into four slots and the DYN segment has eight minislots, and Figure 1(b) presents two communication cycles of the same network in which ST and DYN frames are transmitted. For this network, we consider that four periodic frames $m_1$, $m_2$, $m_3$ and $m_4$ with respective periods one cycle, 2 cycles, 2 cycles and 4 cycles are scheduled to be transmitted in slots 1, 2, 3 and 4 respectively. On the other hand, we consider that four aperiodic frames $m_a$, $m_b$, $m_c$ and $m_d$ with minimum inter-arrival time of 4 cycles are scheduled to be transmitted in DYN slots 1, 4, 2 and 4 respectively. $m_a$ and $m_b$ are transmitted in the first communication cycle, while $m_c$ and $m_d$ are transmitted in the second cycle. $m_a$ and $m_d$ are scheduled in the same DYN slot, so only an instance of the either can be transmitted in each cycle, based on their priorities (see Section II-B below). We can observe that the length of ST slots is fixed and the transmitted frame can occupy all or a part of the slot. In contrast the DYN slots expand to the number of minislots needed to transmit the frames.

In a distributed embedded network, application processes running on different nodes exchange messages packed into frames of fixed format by the Communication Controller. A frame may carry only a part of a message or a multiplicity of messages. In the sequel, we consider without loss of generality that every frame carries only one message, so the response time of a message is the same as the response time of the frame that carries it. Details on FlexRay frame format and the associated maximum transmission time $C_m$ of a FlexRay frame $m$ can be found in [15], [16].

B. FlexRay protocol messages parameters

The designer of a FlexRay communication system has to decide its parameters which are the length of the ST segment (which we refer as $T_{ST}$), the length of the DYN segment (which we refer as $T_{DYN}$) which equals the length of a minislot (which we refer as $T_{MS}$) times the total number of minislots (which we refer as $N_{MS}$), and the lengths of SW and NIT. As our analysis concerns the DYN segment, the other segments ST, SW and NIT can conceptually be viewed as a single combined segment, and with no loss of generality, we consider $T_{ST}$ to be the length of this single combined segment. Also we let $T_{bus}$ denote the combined length of all four segments, i.e., the length of one full bus cycle. The designer also has to decide the assignment of frame identifiers
to nodes and messages, and the priority ordering of the DYN messages having the same frame identifier. Another parameter to be decided is $p_{\text{LatestTx}_N}$, that indicates the last minislot in which node $N_i$ can start transmitting a frame. $p_{\text{LatestTx}_N}$ is determined from the largest DYN frame sent by node $N_i$. If node $N_i$ generates a message $m$, then we denote the value $p_{\text{LatestTx}_N}$ as $T_m^{\text{last}}$ (which is the last minislot of a DYN segment in which message $m$ can be transmitted).

In the sequel, we denote by $ID_m$ the frame identifier of a DYN message $m$ (it specifies the minislot in which the transmission of $m$ can start). A DYN message $m'$ has a higher priority than a DYN message $m$ if either $m'$ has a lower frame identifier than $m$, i.e. $ID_{m'} < ID_m$, or $ID_{m'} = ID_m$ and $m'$ is assigned a higher priority than $m$. We denote by $\lfloor m \rfloor$ the set of messages having lower frame identifier than $m$ (and hence higher priority than $m$), by $hp(m)$ the set of messages having the same frame identifier as $m$ but of higher priority. Let $\text{dom}(m) = \lfloor m \rfloor \cup hp(m)$ denote the priority ordered list of all the messages that dominate $m$ in priority.

The structure of the FlexRay protocol puts constraints on a set of messages that can be transmitted in a single cycle as discussed next. For a message $m$, let $\phi_m$ denote the number of extra minislots (in addition to its assigned minislot) required for the communication of $m$, i.e. $\phi_m = \lceil C_m / T_{MS} \rceil - 1$, where $\lceil C_m / T_{MS} \rceil$ is the communication time of $m$ in the units of $T_{MS}$. When a DYN message $m$ is transmitted, the transmission minislot position for the messages with identifiers greater than $ID_m$ is shifted by the value $\phi_m$. Therefore, after the transmission of a set of messages $M \subseteq \text{dom}(m)$ of higher priority messages in a cycle, the transmission of a message $m$ in the same cycle can start in minislot $ID_m + \sum_{m \in M} \phi_m$. Moreover, this value must be smaller than $T_m^{\text{last}}$, and otherwise $m$ cannot fit in the same cycle as messages in $M$. Thus we have the following definition of the transmissibility of a set of messages in a cycle.

**Definition 1** (Transmissible messages)

A message $m$ with $\text{dom}(m) = \emptyset$ (i.e., $m$ is dominated by no other message) is transmissible in a cycle if $ID_m \leq T_m^{\text{last}}$. A message $m$ and a set of higher priority messages $M \subseteq \text{dom}(m)$ are transmissible in one cycle if $M$ is transmissible in one cycle and:

$$\forall m' \in M : ID_{m'} \neq ID_m \land \left[ ID_m + \sum_{m' \in M} \phi_{m'} \leq T_m^{\text{last}} \right] .$$ (1)

When $M$ is transmissible in a cycle $j$, and $M \cup \{m\}$ is not transmissible in that cycle, we say that $j$ is filled by $M$ for $m$.

### III. RESPONSE TIME ANALYSIS OF FLEXRAY MESSAGES

#### A. Problem statement

Consider the set $D$ of DYN frames in a FlexRay network (every frame is messages), where every message $m \in D$ has:

- A length $L_m$ that leads to a communication time $C_m$ and extra minislots $\phi_m = \lceil C_m / T_{MS} \rceil - 1$.
- A set $\text{dom}(m)$ of messages that dominate $m$ in terms of priority as determined by frame identifier $ID_m$ and priority ordering $hp(m)$ among the messages of the same identifier.
- A minimum interarrival time $T_m$, where the arrival time of an instance of $m$ is defined as the time the instance of $m$ is ready to be transmitted on the FlexRay bus by the CC of its sending node.
- The value $T_m^{\text{last}}$ of the last minislot in which $m$ can be transmitted.

For schedulability analysis, we want to compute the worst-case response time $R_m$ of $m$, i.e. the worst possible delay from the arrival time of an instance of $m$ to the moment all the bits of $m$ are received by the CC of its receiving node.

#### B. Response time decomposition of FlexRay messages

The end-to-end delay $D_m$ of a FlexRay message $m$ is given by:

$$D_m = R_{SP(m)} + R_m + R_{RP(m)};$$ (2)

where:

- $R_{SP(m)}$ is the worst-case response time of the sender-process, i.e., the maximum delay from the moment the transmission of message $m$ is requested to the moment when $m$ is queued in the CC ready for transmission on the bus.
- $R_m$ is the worst-case response time of $m$.
- $R_{RP(m)}$ is the worst-case response time of the receiver process, i.e. the maximum delay from the arrival of the message in the CC of the receiver to the time the receiver process pickups the message from the CC.

$R_{SP(m)}$ and $R_{RP(m)}$ depend entirely on the scheduling of the tasks in the application layers of the sender and receiver, and methods for determining their values are known, see for example [17] for EDF-based scheduling. Our analysis concerns only the delays related to the FlexRay protocol, and so we only focus on the computation of $R_m$.

The worst case response time of a DYN message $m$ can be decomposed as follows (see Figure 2) [12]:

$$R_m = Q_m + C_m = (B_m + W_m) + C_m;$$ (3)

where:
For the third iteration with \( B_m \) is experienced when no message in \( M \) is transmitted in the current bus cycle and message \( m \) arrives just after the beginning of its minislot. Then \( B_m \) is given by the following equation [12]:

\[
B_m = T_{bus} - (T_{ST} + (ID_m - 1) \times T_{MS})
\]

where:
- \( X_m \) is the number of bus cycles for which the transmission of \( m \) is not possible because of transmission of higher priority messages;
- \( W'_{m} \) is the time elapsed in the final cycle in which \( m \) is transmitted.

Our model for the computation of \( R_m \) assumes that \( Q_m \) and \( C_m \) occur ideally following the FlexRay specification. The arrival pattern of messages in \( dom(m) \), that can cause the worst-case interference delay \( W_m \), is the one that fills each bus cycle \( k \leq X_m \) for \( m \) with a set of transmissible messages \( M_k \subseteq dom(m) \) such that the number of consecutively filled cycles is maximized. If \( M \) denotes the set of all collections of higher priority messages that can fill a cycle for \( m \), i.e., \( M = \{ M \subseteq dom(m) | M \text{ is transmissible in one cycle and } \sum_{m \in M} \phi_m + ID_m > T_{last} \} \), then finding \( W_m \) is tantamount to finding a longest sequence of elements of \( M \) that are together feasible (i.e., meet the set of constraints of Subsection III-E below).

C. Conservative analysis of [12]

[12] estimated the worst-case response time by iteratively solving a sequence of ILPs. In the first instance of the ILP, only a single instance of each message is considered for estimating the worst-case response time. If a second instance of certain messages can arrive within this period, then a second instance of ILP is solved by including the additional instances of such messages. The process is repeated until it converges (no additional arrivals of any of the messages is possible in the already estimated response time). It turns out that this iterative process is conservative (i.e., overestimates the worst case response time) since it ignores the constraints on inter-arrival times, as illustrated below. We use the notation \( T_m \) to denote the minimum inter-arrival time of message \( m \).

Consider the setting of Figure 3, where the DYN messages \( m_1, m_2, m_3 \) and \( m \) are assigned the identifiers as illustrated in Figure 3a. We suppose that the length of a cycle is equal to \( 20 \times T_{MS} \) and that \( T_{m_3} = 1.9 \times T_{bus} = 38 \times T_{MS} \). We can observe for this example that \( m \) and \( m_3 \) (or any set of messages including \( m_3 \)) cannot fit in the same cycle (since \( ID_m + \phi_{m_3} = 11 > T_{last} = 9 \), and further \( m \) together with \( m_1 \) and \( m_2 \) cannot fit in the same cycle (since \( ID_m + \phi_{m_3} + \phi_{m_2} = 22 > T_{last} \)). We can also observe that \( \{ m_1, m_2, m_3 \} \) is transmissible in one cycle and thus if an instance of these messages is transmitted in cycle \( k \), its arrival time is no earlier than the beginning of its minislot in cycle \( k - 1 \). This is due to the fact that a released message cannot be delayed beyond a cycle in which it is transmissible.

We consider without loss of generality that the jitters of messages are equal to zero in the sequel. Let us compute \( R_m \) for \( m \) of Figure 3 using the method of [12]. To be short, the first cycle where message \( m \) arrived and missed its slot is not shown in Figures 3b-e, and so the blocking time \( B_m = 5T_{MS} \) has to be added to the illustrated part to obtain \( R_m \).

- For the first iteration with \( I_m(m) = \{ m_1, m_2, m_3 \} \), we obtain \( R^0_m = 3T_{bus} + C_m \) decomposed as follows: \( B_m = 5T_{MS} \), \( W_m = 2T_{bus} \) and \( W'_m = T_{ST} + 6T_{MS} \), so that \( B_m + W'_m = T_{bus} \), and \( Q_m \) is transmissible in one cycle and thus if an instance of these messages is transmitted in cycle \( k \), its arrival time is no earlier than the beginning of its minislot in cycle \( k - 1 \). This is due to the fact that a released message cannot be delayed beyond a cycle in which it is transmissible.

It turns out that the message transmission pattern illustrated in Figure 3d is not possible in reality due to the minimum inter-arrival time of messages. Indeed, a new instance of \( m_3 \) cannot be ready in the fourth cycle before its minislot because:

(i) the instance of \( m_3 \) transmitted in the third cycle must have arrived after its minislot in the second cycle (otherwise \( m_3 \) would have been transmitted in the second cycle), and
(ii) the second instance of \( m_3 \) satisfying its minimum inter-arrival time can arrive in the fourth cycle only after the minislot
that possible in reality. Note this conservative solution also means 
value of $R$ transmission in [13]) that requires the 
D. Optimistic analysis of [13]

In [13], another ILP formulation that attempts to overcome 
the shortcoming of [12] by introducing a constraint (see (6) 
in [13]) that requires the transmission of $n$ instances of every 
message $m_i$ within $k$ consecutive cycles to be allowed if $n \leq 
[k \times T_{bus}/T_i]$. Note this is a constraint on inter departure 
times (rather on inter arrival times) and it turns out that such 
a formulation leads to optimistic results (less than the precise 
value) because the number of instances of a message satisfying 
the ILP constraint of [13] may be less than what is possible. 
The reason being that constraint (6) of [13] does not consider 
the delay caused by non-transmissibility of messages due to 
the presence of filled cycles (it only considers the delay caused 
by the separation in the inter arrival times). For example, if 
we modify the parameters of the bus or the messages in the 
example of Figure 3 such that $m_3$ and $\{m_1, m_2\}$ are no longer 
transmissible in the same cycle (say by letting $T_{max}^{m_3} = 6$), the 
exact value of $X_m$ will be $6T_{bus}$, but the ILP of [13] will 
compute an optimistic result, namely, $X_m = 4T_{bus}$. Since a 
lower bound on response time is not useful for any real-time 
schedulability analysis, we did not simulate the formulation 
of [13] to compare against our formulation or that of [12]. 
Furthermore, the ILP formulation of [13] does not consider the 
case of multiple messages assigned the same frame identifier.

E. Precise computation of the value of $W_m$

Next we propose an ILP formulation that accurately takes 
to account the requirements on minimum inter-arrival times to 
obtain the precise value for $W_m$. Another significantly 
important aspect of our formulation is that only two instances 
of ILPs that compute $X_m$ and $W_m'$, respectively are performed, 
contrary to the formulation in [12] in which a sequence of ILPs 
are executed iteratively multiple times.

Given the set of messages $\text{dom}(m) = 
\{m_1, \ldots, m_i, \ldots, m_{p-1}\}$ ordered in decreasing priority 
and $m_p = m$, the objective of our first ILP formulation is to 
compute $X_m (= W_m/T_{bus})$, i.e., maximize the number of 
filled cycles for $m$. First, we determine an upper bound $R_m$ 
for the response time $R_m$ beyond which the message $m$ is
deemed unschedulable: $\overline{R_m}$ is taken to be either the deadline of $m$ or an upper bound computed by the heuristic proposed in [18], which ever is smaller. Then an upper bound $U$ for $X_m$ is given by $U := \lceil \frac{\overline{R_m}}{T_{bus}} \rceil$, and so the ILP maximization need not explore for a solution beyond this upper bound (as we have captured in the objective function (16)). (If the result of ILP maximization equals this upper bound, then this simply means that $m$ is unschedulable.) In the sequel, for simplicity we use index $i$ to signify message $m_i$, for example $ID_i$ represents $ID_{m_i}$.

We introduce a binary constant $f_{ik}$ to designate whether messages $i$ and $k$ have the same identifier: $f_{ik} = 1$ iff $ID_i = ID_k$, for $i, k = 1, \ldots, p$. The binary decision variable $x_{ij}^l$ is introduced to take the value 1 if message $i$ is transmitted in cycle $j$, and the binary decision variable $u_i^l$ is set to 1 iff message $i$ is not transmissible in cycle $j$. Note our only decision variable $x_{ij}^l$ is indexed by a message index $i$ as well as a cycle index $j$, meaning multiple instances of the same message departing in multiple cycles has simultaneously been considered in our formulation allowing us to formulate the problem as a single instance of ILP (as opposed to a sequence of instances of ILP as in [12]).

The following equation captures the non-transmissibility condition and defines $u_i^l$:

$$ u_i^l = 1 \Rightarrow \left[ \sum_{k < i} x_{k,i}^l \phi_k + ID_i > T_i^{last} \right] \vee \left[ \sum_{k < i} x_{k,i}^l f_{ik} > 0 \right], \forall i, j. \tag{6} $$

The first conjunct states that the messages with lower identifier fill the bus beyond the last time $m_i$ can be sent, whereas the second conjunct states that a message in $hp(m_i)$ is sent in the same cycle.

(6) can be expressed by linear equations as follows. The following equation captures the forward implication of (6):

$$ \left[ \sum_{k < i} x_{k,i}^l \phi_k + ID_i \right] > \left[ (u_i^l - \sum_{k < i} x_{k,i}^l f_{ik}) T_i^{last} \right], \forall i, j; \tag{7} $$

and the following two equations capture the backward implication of (6):

$$ \left[ \sum_{k < i} x_{k,i}^l \phi_k + ID_i \right] \leq \left[ (1 - u_i^l) T_i^{last} + 2u_i^l \times N_{MS} \right], \forall i, j; \tag{8} $$

$$ \sum_{k < i} x_{k,i}^l f_{ik} - u_i^l \leq 0, \forall i, j. \tag{9} $$

Next, since in any cycle, only transmissible messages can be transmitted (property of the protocol), we have the constraint:

$$ x_i^l \leq 1 - u_i^l, \forall i, j. \tag{10} $$

Next to capture the constraint of minimum inter-arrival times of messages, consider a cycle $j$, and suppose that a message $i$ is transmissible in this cycle ($u_i^l = 0$) but no instance of $i$ is transmitted, and $q \geq 2$ instances of message $i$ are transmitted in the next $k$ cycles, ($0 = x_i^l = (\sum_{l=0}^{k-1} x_i^{l+j+1} - q) = (x_i^{j+k} - 1$)). See for illustration Figure 4 (same as case c of Figure 3) where $m3$ is transmissible in cycle 2, and two copies of message $m3$ are transmitted in next two cycles 3 and 4. Then the first copy of message $i$ must have arrived after the beginning of the minislot of cycle $j$ in which it could be transmitted (recall message $i$ is transmissible in cycle $j$ but didn’t get transmitted in that cycle: $u_i^l = x_i^l = 0$), and the last ($q$th) copy of message $i$ must have arrived before the beginning of the minislot of cycle $j + k$ in which it got transmitted (recall there are a total of $q$ transmissions in a total of $k$ cycles beyond the cycle $j$). Thus if we let $\Delta_{i,k}^{j,k}$ denote the time-interval between the two minislots of cycles $j$ and $j + k$ in which message $i$ could be got transmitted (see Figure 4 and (12)), then this interval must be long enough to accommodate the arrival of $q$ copies of message $i$, i.e., $\Delta_{i,k}^{j,k} > (q - 1) T_i = (\sum_{l=0}^{k-1} x_i^{j+l} - 1) T_i$, where $T_i$ is the minimum inter-arrival time of message $i$. Thus the following constraint must hold:

$$ \left[ u_i^l = 0 \right] \land \left[ x_i^l = 0 \right] \Rightarrow \left[ \sum_{l=0}^{k} x_i^{j+l} - 1 \right] T_i, \forall i, j, k; \tag{11} $$

where

$$ \Delta_{i,k}^{j,k} := k \times T_{bus} + \left[ \sum_{n=1}^{j-1} x_i^{j+n-1} \phi_n \right] \times T_{MS}. \tag{12} $$

Note $\Delta_{i,k}^{j,k}$ is the interval spanning the instances in $j$th and the $(j + k)$th cycles when the message $i$ can be transmitted, accounting for the transmission times of higher priority messages ($n < i$) that were transmitted in those cycles (captured by $x_{n,i}^j$ and $x_{n,i}^{j+k}$ variables). (11) can be written in a linear form as follows:

$$ \left[ \Delta_{i,k}^{j,k} - (\sum_{l=0}^{k} x_i^{j+l} - 1) T_i \right] \geq \left[ u_i^l + x_i^l \right] $$

$$ \times \left[ k T_{bus} - \sum_{n=1}^{i} \phi_n - (k - 1) T_i \right] / 2, \forall i, j, k. \tag{13} $$

(13) captures the implication of (11) when $u_i^l = x_i^l = 0$, and trivially holds otherwise.

Also, the minimum inter-arrival time constraint should hold for any sequence of arrivals (and not just for the ones that happen to occur after a cycle-$j$ where $u_i^l = x_i^l = 0$). To accommodate this, we introduce an empty cycle 0 and allow $j$ to take the value 0, and set:

$$ x_i^0 = u_i^0 = 0, \forall i. \tag{14} $$

(14) also ensures that the worst-case blocking time is already experienced prior to experiencing the worst-case interference delay, guaranteeing that the worst-case response time computed by the ILP arises from a feasible scenario and thus it is exact.

For the example of Figure 4, we can check that the two instances of $m3$ in cycle 1 and 3 satisfy the minimum inter-arrival time as $\Delta_{3,3}^{3,3} > T_{m3}$ but the two instances of $m3$ in cycle 3 and 4 do not satisfy the minimum inter-arrival time as $\Delta_{3,2}^{2,2} < T_{m3}$. 
Next, for maximizing the filled cycles for message $m_p$, its filled cycles must be contiguous:

$$[u^j_p \geq u^{j+1}_p], \quad \forall j > 0.$$  \hspace{1cm} (15)

Finally the objective function of the ILP for computing the precise value of $X_m$ is:

$$\text{Maximize } \sum_{j=1, \ldots, U} u^j_p, \quad (16)$$

where $U$ is the upper bound of $X_m$ as determined in the beginning of this subsection. The objective function given by (16) and the set of constraints given by (6)-(15) form an instance of an ILP program. The result of the maximization gives the maximum number of cycles $X_m$ that can be filled by messages in $\text{dom}(m)$ while respecting the minimum interarrival times.

To obtain $W'_m$, we can modify the objective function of the above ILP program as:

$$\text{Maximize } \sum_{i=1}^{p-1} x_i^{m+1} \phi_i, \quad (17)$$

to maximize the additional communication time of higher priority messages sent in cycle $X_m + 1$ in which the message $m$ can also be sent, and add an additional constraint that forces the first $X_m$ cycles to be filled for $m$:

$$u^j_p = 1, \quad \forall 0 < j \leq X_m. \quad (18)$$

Let us illustrate our ILP solution with the simple example of Figure 3. For this example, we have $R_m = T_m = 5.5T_{bus}$ and then the upper bound of $X_m$ is $6T_{bus}$. The solution of our ILP for computing $X_m$ that satisfies all the constraints is such that $x_3^1 = x_3^2 = x_4^1 = 1$ and $x_3^1 = x_3^2 = 1$ which implies $u^j_p = 1, j = 1, ..., 4$ (we suppose $p = 4$ and $m = m_p = m_4$) and $X_m = 4$. This assignment of variables satisfies constraints (6)-(15), in particular constraint (11), as $\Delta_3^{0,2} = 2T_{bus} > T_3 = 1.9T_{bus}$. Contrary to [12], our ILP solution will not accept assignment of variables leading to the message transmission pattern of Figure 3d because it does not satisfy constraint (11), as $\Delta_3^{2,2} = 1.8T_{bus} < T_3 = 1.9T_{bus}$. In the same way and contrary to the solution of [12], our ILP solution will not accept the message transmission pattern of Figure 3e, which gives $X_m = 5$, because it does not satisfy constraint (11), as $\Delta_4^{1,4} = 3T_{bus} < T_3 = 4T_{bus}$. The final solution $X_m = 6$ of [12] for this example is not accepted by our ILP for similar reasons. Thus for the Example of Figure 3, the precise worst-case response time contains two less filled cycles compared to the result obtained using the ILP formulation of [12].

Remark 1: [18] allows arbitrary message deadlines unrelated to their periods, and so a prior instance of $m_p$ may contribute in the interference delay of the current instance. Therefore, [18] approaches the computation of $R_{m_p}$ by computing iteratively the busy period of level-$p$ within which all possible instances of $m_p$ are taken into account, using the same ILP formulation of [12]. Therefore, the approach of [18] is doubly iterative (iterations to account for all possible instances of messages in $\text{dom}(m_p)$ and iterations to account for all possible instances of $m_p$ in the busy period). In the current formulation, we assume that the deadline of $m_p$ is less than its minimum interarrival time $T_p$ and so different instances of $m_p$ don’t interfere with each other. Our approach however is general enough to account for all instances of $m_p$ that may have arrived in the busy period of level-$p$ without needing to iterate over multiple instances of ILP. Indeed, prior instances of $m_p$ can be taken into account the same way as we account for the multiple instances of other messages, namely, by introducing the variable $x^j_p$ (which takes the value one iff an instance of message $m_p$ is transmitted in cycle $j$). The instances of $m_p$ satisfy the same constraints as the other messages (such as inter arrival times) and the older instances take priority over the newer instances of $m_p$ (while the priority ordering with respect to the other messages remains the same as prescribed).

F. Correctness of our ILP formulation

In our formulation, we don’t have any explicit constraint on arrival times of messages. Instead, the constraint (11) of our ILP captures a constraint on departures ($x^j_i$ are departure variables). To establish correctness, we show that the result of the ILP proposed above is the same as a formulation that explicitly captures the interarrival time constraints. Since arrival times are real-valued, this alternative formulation is no longer an ILP rather a mixed-ILP (MILP), and so clearly the ILP formulation is preferable owing to its computational advantages over the alternative MILP formulation (and this is our motivation for proposing the ILP formulation in the first place).

First, letting $a^r_i$ and $a^s_i$ denote arrival times of two successive instances of message $m_i$ for which departures occur in cycles $r$ and $s$ (so that $x^s_i = x^s_i = 1$), $a^r_i$ and $a^s_i$ must satisfy the minimum interarrival time constraint. This is captured by the following equation:

$$\forall i, r < s : [x^r_i = x^s_i = 1] \Rightarrow [a^r_i \leq a^s_i - T_i], \quad (19)$$

which can be written in a linear form as follows:

$$a^r_i - a^s_i - T_i \geq (x^r_i + x^s_i - 2) \times T_i, \forall i, r < s.$$
Next, to simplify notation, let $d_i^j$ denote the time at which the transmission of a message $m_i$ may start in a cycle $j$, i.e.

$$d_i^j = (j-1) \times T_{bus} + T_{ST} + \left( \sum_{n<i} x^j_n \times \phi_n + ID_i \right) \times T_{MS}.$$ 

Since the arrival time $a_i^r$ must be lower than the corresponding departure time, we also have:

$$a_i^r < d_i^r, \forall i, r.$$  

Finally, a feature of FlexRay is that a message cannot be delayed beyond a cycle in which it is transmissible, i.e. if ever there is a cycle where the message is transmissible but without any transmission, then an arrival of the message must not have happened yet. This is equivalent to saying that if $j$ is a cycle preceding a cycle $r$ and is empty for $m_i$ (meaning $m_i$ is not transmissible in $j$, $u_i^j = 0$, but not transmitted, $x_i^j = 0$), then the instance of message $m_i$ that departs in cycle $r$ must have arrived to miss cycle $j$ (otherwise it would depart in cycle $j$ and not in cycle $r$). This is captured by the following constraint:

$$\forall j < r: [u_i^j = x_i^j = 0] \Rightarrow [a_i^r \geq d_i^j]. \quad (21)$$

The above can be linearized and written as follows:

$$a_i^r \geq d_i^j - j \times T_{bus} \times (u_i^j + x_i^j), \forall i, j < r.$$ 

By replacing (11) with (19)-(21) in our ILP of Section III-E, we obtain a MILP that explicitly captures the correct set of constraints on the interarrival times (and the corresponding departure times). We show that the ILP and the MILP formulations yield the same maximization result, thereby proving the correctness of the ILP formulation. We first show that whenever the arrival sequences satisfy (19)-(21), then the departure sequences satisfy (11).

**Lemma 1**: (19)-(21) imply (11), i.e., the following holds for all message index $i$:

$$\forall r < s \text{ s.t. } [x_i^r = x_i^s = 1]: [a_i^r \leq a_i^s - T_i]$$

$$\land \left[ \forall j < r \text{ s.t. } [u_i^j = x_i^j = 0]: d_i^j \leq a_i^r < d_i^s \right] \Rightarrow (22)$$

$$\forall j, k: [u_i^j = 1] \lor [x_i^j = 1] \lor \left[ \Delta_{j,k} > (\sum_{l=0}^{k} x_i^{j+l-1}) T_i \right]. \quad \square$$

**Proof**: We need to show that if (19)-(21) are satisfied, then (11) is satisfied. As shown in Figure 5a, let two successive instances of message $m_i$ be transmitted in cycles $r$ and $s$ respectively, so that $x_i^r = x_i^s = 1$; and $j$ be the first cycle before cycle $r$ where $m_i$ is transmissible but no instance of $m_i$ is transmitted, i.e. $[u_i^j = 0] \land [x_i^j = 0]$. Thus for every cycle $j'$ such that $j < j' < r$, we have $[u_i^{j'} = 1] \lor [x_i^{j'} = 1]$. Figure 5a depicts the situation when the two arrivals satisfy constraints (19)-(21). Clearly as can be seen from Figure 5a, for every $k \geq s-j$, we have $\Delta_{j,k} > (\sum_{l=0}^{k} x_i^{j+l-1}) T_i$, as $\sum_{l=0}^{k} x_i^{j+l-1} = 0$ if $r-j \leq k < s-j$ and is equal to $-1$ otherwise. We can generalize the case illustrated by Figure 5a to any number $q$ of successive arrivals of $m_i$ as illustrated by Figure 5b as follows: If any pair of successive arrivals times $(a_i^q, a_i^q)$ of the instances of $m_i$ transmitted in the next $k$ cycles beyond cycle $j$, i.e. $q = \sum_{l=0}^{k} x_i^{j+l-1}$ instances, satisfies constraints (19)-(21), we have $\Delta_{j,k} > (\sum_{l=0}^{k} x_i^{j+l-1}) T_i$, because the interval $a_i^q - a_i^q$ is contained in $\Delta_{j,k}$ and $a_i^q - a_i^q \geq (q-1) T_i$.

Letting $R_{MP}^{ILP}$ and $R_{MP}^{MILP}$ denote the response times computed by the ILP and MILP formulations respectively, one implication of Lemma 1 is that $R_{MP}^{ILP} \geq R_{MP}^{MILP}$ since by Lemma 1, ILP constraints are weaker than the MILP constraints and hence the ILP formulation yields a larger maximum. To complete the correctness proof, it remains to establish the converse. For this it suffices to show that when-
ever a departure sequence satisfies (11), then a corresponding arrival sequence satisfies (19)-(21). This would then imply that corresponding to every candidate solution of ILP, there is a corresponding candidate solution of MILP (that has the same departure sequence as the ILP candidate solution and so has the same response time as the ILP candidate solution), and so the result of MILP maximization (over all its candidate solutions) can only be larger.

**Lemma 2:** For every candidate solution satisfying (11), there exists a candidate solution satisfying (19)-(21), i.e., the following holds for all message index $i$:

\[
\forall j, k : \left[ u_i^j = 1 \right] \lor \left[ x_i^j = 1 \right] \lor \left[ \Delta_i^{j,k} > \left( \sum_{l=0}^{k} x_i^{j+l} - 1 \right) T_i \right] \Rightarrow \\
\forall r < s \text{ s.t. } \left[ x_i^r = x_i^s = 1 \right] : \exists \left( a_i^r, a_i^s \right) \text{ s.t. } a_i^r \leq a_i^s - T_i \\
\land \forall j < r \text{ s.t. } \left[ a_i^j = x_i^j = 0 \right] : d_i^j \leq a_i^r < d_i^s. \\
(23)
\]

**Proof:** We need to find $a_i^r, a_i^s$ satisfying (19)-(21), that is the right hand side of (23), given that (11) is satisfied, that is the left hand side of (23). As shown in Figure 5c, consider two successive instances of message $m_i$ transmitted in cycles $r$ and $s$, so that $x_i^r = x_i^s = 1$; and $j$ be the first cycle before cycle $r$ where $m_i$ is transmissible but no instance of $m_i$ is transmitted, i.e. $u_i^t = x_i^t = 0$. Thus for every cycle $j$ such that $j < j' < r$, we have $\left[ u_i^j = 1 \right] \lor \left[ x_i^j = 1 \right]$. Clearly as can be seen from Figure 5c, if the two departures in cycles $r$ and $s$ satisfy (11), i.e. $\Delta_i^{j,k} > T_i$, then there exists $a_i^r$ and $a_i^s$ such that $d_i^j \leq a_i^r < d_i^s$ and $d_i^j \leq a_i^s < d_i^s$ (or, if there exists $t, r < t < s$ such that $[a_i^t = x_i^t = 0]$; $d_i^t \leq a_i^s < d_i^t$), and $a_i^s \geq a_i^r + T_i$, because the interval $\left[ a_i^r, a_i^s - T_i \right]$ is empty (i.e. $a_i^r$ will be within the interval $[d_i^j, a_i^s - T_i]$ and $a_i^s$ within $[a_i^r + T_i, d_i^s]$). We can generalize the case illustrated by Figure 5c to any number $q$ of departures of $m_i$ in the next $k$ cycles beyond cycle $j$ as follows: If $\Delta_i^{j,k} > (q-1)T_i$, where $q = \sum_{l=0}^{k} x_i^{j+l}$, then clearly the interval $\Delta_i^{j,k}$ is big enough to accommodate $q$ arrivals $a_i^1, .., a_i^q$ satisfying (19), as depicted by Figure 5b. Next, for every pair of departure cycles $(r, s)$ such that $r < s \leq j + k$, (11) implies $\Delta_i^{j,k} > (\sum_{l=0}^{k} x_i^{j+l} - 1) T_i$ and thus $\delta_i^j(r, s)$ is not empty, so there exist arrival times $a_i^r < d_i^s$ and $a_i^s < d_i^s$, i.e. that satisfy (20). Finally, all the arrival times considered above for the corresponding departure times are beyond cycle $j$, so they satisfy (21).

Lemmas 1 and 2 together imply the equality of response times computed by the ILP versus MILP formulations as formalized in the following theorem.

**Theorem 1** Letting $R_{m}^{ILP}$ and $R_{m}^{MILP}$ denote the response time of message $m$ computed with the ILP and the MILP respectively, it holds that:

\[
R_{m}^{ILP} = R_{m}^{MILP}.
\]

**Proof:** From Lemma 1, (19)-(21) is stronger than (11), and so MILP will yield a smaller maximum value compared to ILP, i.e., $R_{m}^{MILP} \leq R_{m}^{ILP}$. Conversely from Lemma 2, every candidate departure sequence solution satisfying (11) has a corresponding candidate arrival sequence solution satisfying (19)-(21) possessing the same said departure sequence, and hence the same value of response time (recall that response time is determined by the number of filled cycles and hence is a function of only the departure sequence, and so same departure sequence means the same response time). This means that every departure sequence solution satisfying (11) has a corresponding arrival sequence satisfying (19)-(21) such that they yield the same response time. Then the result of MILP maximization can only be higher, i.e., $R_{m}^{MILP} \geq R_{m}^{ILP}$, completing the proof.

**Remark 2:** While the MILP formulation with constraints (19)-(21) also computes the precise worst-case response time $R_{m}$, the ILP formulation is preferred because of the smaller number of variables and constraints. We have been able to confirm this experimentally by also implementing and simulating the MILP for the benchmark (see results in Table III where on an average, the ILP outperforms the MILP, with only two cases of exception).

IV. APPLICATION TO A BENCHMARK AND DISCUSSION

We evaluate our methodology by applying it to a benchmark problem of automotive applications. For our tests we use the Society of Automotive Engineers (SAE) benchmark [19] that provides a set of signals (i.e., messages of electronic control units (ECUs)) for a prototype electric car and that serves as a good example to illustrate a FlexRay network application. The benchmark consists of 7 ECUs that are: the batteries (BAT), the vehicle controller (V/C), the inverter/motor controller (I/M/C), the instrument panel display (I/P/D), driver inputs (D/I), brakes (BRK), and the transmission control (T/C). The 7 ECUs exchange messages of a total of 53 signals consisting of 22 periodic signals and 31 sporadic signals. As periodic message are scheduled in the ST segment, we consider only the 31 sporadic messages and schedule them in the DYn segment. We consider that every message is packed into one frame.

To evaluate the performance of our approach, we consider three different parameter-configurations (denoted cfg1, cfg2 and cfg3 in Table I) for the FlexRay network. These three configurations differ in cycle, ST and DYn segments lengths. This is to cover different cases of schedulability of the messages in the experimental study.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$T_{bus}$ (MT)</th>
<th>$T_{ST}$ (MT)</th>
<th>$T_{MS}$ (MT)</th>
<th>$N_{MS}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfg1</td>
<td>170</td>
<td>60</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>cfg2</td>
<td>120</td>
<td>40</td>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>cfg3</td>
<td>150</td>
<td>30</td>
<td>2</td>
<td>60</td>
</tr>
</tbody>
</table>

Table II shows the set of 31 sporadic messages (in the same order as appearing in the benchmark) and their parameters where:
Table II

<table>
<thead>
<tr>
<th>No</th>
<th>Message ID</th>
<th>(T_{\text{last}}) (MT)</th>
<th>(T) (MT)</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14 7</td>
<td>40 30 50</td>
<td>1000 720 1400</td>
<td>BAT</td>
<td>V/C</td>
</tr>
<tr>
<td>2</td>
<td>15 4</td>
<td>45 35 55</td>
<td>800 720 1400</td>
<td>D/I</td>
<td>V/C</td>
</tr>
<tr>
<td>3</td>
<td>16 3</td>
<td>45 35 55</td>
<td>800 720 1400</td>
<td>D/I</td>
<td>V/C</td>
</tr>
<tr>
<td>4</td>
<td>17 5</td>
<td>45 35 55</td>
<td>800 720 1400</td>
<td>D/I</td>
<td>V/C</td>
</tr>
<tr>
<td>5</td>
<td>18 4</td>
<td>47 37 57</td>
<td>900 720 1400</td>
<td>BRK</td>
<td>V/C</td>
</tr>
<tr>
<td>6</td>
<td>19 4</td>
<td>45 35 55</td>
<td>800 850 1400</td>
<td>D/I</td>
<td>V/C</td>
</tr>
<tr>
<td>7</td>
<td>20 6</td>
<td>45 35 55</td>
<td>800 850 1400</td>
<td>D/I</td>
<td>V/C</td>
</tr>
<tr>
<td>8</td>
<td>22 6</td>
<td>45 35 55</td>
<td>800 850 1400</td>
<td>D/I</td>
<td>V/C</td>
</tr>
<tr>
<td>9</td>
<td>23 4</td>
<td>40 30 50</td>
<td>1000 1500 1400</td>
<td>BAT</td>
<td>V/C</td>
</tr>
<tr>
<td>10</td>
<td>24 4</td>
<td>40 30 50</td>
<td>1000 1500 1400</td>
<td>BAT</td>
<td>V/C</td>
</tr>
<tr>
<td>11</td>
<td>25 4</td>
<td>40 30 50</td>
<td>1000 1500 1400</td>
<td>BAT</td>
<td>V/C</td>
</tr>
<tr>
<td>12</td>
<td>26 4</td>
<td>45 35 55</td>
<td>800 1500 1400</td>
<td>D/I</td>
<td>V/C</td>
</tr>
<tr>
<td>13</td>
<td>27 4</td>
<td>45 35 55</td>
<td>800 1500 1400</td>
<td>D/I</td>
<td>V/C</td>
</tr>
<tr>
<td>14</td>
<td>28 4</td>
<td>40 30 50</td>
<td>1000 1500 1400</td>
<td>BAT</td>
<td>V/C</td>
</tr>
<tr>
<td>15</td>
<td>31 5</td>
<td>46 36 56</td>
<td>700 1500 1400</td>
<td>V/C</td>
<td>T/C</td>
</tr>
<tr>
<td>16</td>
<td>34 11</td>
<td>40 30 50</td>
<td>1000 1500 1400</td>
<td>V/C</td>
<td>BAT</td>
</tr>
<tr>
<td>17</td>
<td>35 4</td>
<td>40 30 50</td>
<td>1000 1500 1400</td>
<td>V/C</td>
<td>BAT</td>
</tr>
<tr>
<td>18</td>
<td>37 4</td>
<td>47 37 57</td>
<td>900 1500 1400</td>
<td>V/C</td>
<td>BRK</td>
</tr>
<tr>
<td>19</td>
<td>38 4</td>
<td>47 37 57</td>
<td>900 1500 1400</td>
<td>V/C</td>
<td>BRK</td>
</tr>
<tr>
<td>20</td>
<td>39 10</td>
<td>41 31 51</td>
<td>1200 1400 1400</td>
<td>V/C</td>
<td>I/P/D</td>
</tr>
<tr>
<td>21</td>
<td>40 4</td>
<td>47 37 57</td>
<td>2000 1500 1400</td>
<td>V/C</td>
<td>I/M/C</td>
</tr>
<tr>
<td>22</td>
<td>41 4</td>
<td>40 30 50</td>
<td>2000 1500 1400</td>
<td>I/M/C</td>
<td>V/C</td>
</tr>
<tr>
<td>23</td>
<td>44 4</td>
<td>47 37 57</td>
<td>2000 1500 1400</td>
<td>V/C</td>
<td>I/M/C</td>
</tr>
<tr>
<td>24</td>
<td>45 4</td>
<td>40 30 50</td>
<td>2000 1500 1400</td>
<td>I/M/C</td>
<td>V/C</td>
</tr>
<tr>
<td>25</td>
<td>46 4</td>
<td>47 37 57</td>
<td>2000 1500 1400</td>
<td>V/C</td>
<td>I/M/C</td>
</tr>
<tr>
<td>26</td>
<td>47 4</td>
<td>40 30 50</td>
<td>2000 1500 1400</td>
<td>I/M/C</td>
<td>V/C</td>
</tr>
<tr>
<td>27</td>
<td>48 4</td>
<td>47 37 57</td>
<td>2000 1500 1400</td>
<td>I/M/C</td>
<td>V/C</td>
</tr>
<tr>
<td>28</td>
<td>50 5</td>
<td>40 30 50</td>
<td>2000 1500 1400</td>
<td>I/M/C</td>
<td>V/C</td>
</tr>
<tr>
<td>29</td>
<td>51 4</td>
<td>40 30 50</td>
<td>2000 1500 1400</td>
<td>I/M/C</td>
<td>V/C</td>
</tr>
<tr>
<td>30</td>
<td>52 4</td>
<td>40 30 50</td>
<td>2000 1500 1400</td>
<td>I/M/C</td>
<td>V/C</td>
</tr>
<tr>
<td>31</td>
<td>53 4</td>
<td>40 26 57</td>
<td>2000 1600 1400</td>
<td>V/C</td>
<td>I/M/C</td>
</tr>
</tbody>
</table>

*Message number in the benchmark.

- We consider the communication time \( C \) in the unit of number of minislots \( \lfloor C / T_{\text{MS}} \rfloor \) of every message to be equal to the number of bits of the messages plus 2 (to account for the protocol overhead);
- We assign frame identifiers and priorities to messages;
- We define the parameter \( T_{\text{last}} \) for every ECU (based on the size of the largest frame sent by this ECU), and subsequently for every message of the ECU;
- We assign a minimum inter-arrival time \( T \) to every message, that we consider as the message deadline (no inter-arrival time of sporadic messages is given in the benchmark and so we defined our own values).

We used for our simulation CPLEX 12.2 as ILP solver, and AMPL as modeling language. The simulation is carried out on an AMD Athlon X2 Dual-core 2.10Ghz processor with 3GB of RAM. In Table III we present our simulation results for DYN messages 20-31 for the three configurations. We record the precise worst-case response times and their solver times (given by the AMPL timing parameter _total_solve_time) obtained with our ILP of Section III-E (ILP) as well as the corresponding results obtained using the method of [12], for each of the three configurations. We also record the solver time for the MILP formulation of Section III-F (the response times for the MILP are identical to those of our ILP). Cells of Table III with no numerical value are those where the solver doesn’t stop after running for more than one hour, while cells...
In [12], the problem of computing $\sum_{i=1}^{U} x_i^2 \leq 1$, $\forall i$, under the setting of $cfg3$. Indeed if we drop (12)-(13) from our ILP and add a constraint enforcing that only one instance of every message can be transmitted ($\sum_{j=1}^{U} x_j \leq 1$, $\forall i$), we obtain similar or better solver time than the method of [12] (for example, the solver time for computing the response time of message 31 in $cfg3$ with the above mentioned modification reduces from 8.75 seconds to 3.77 seconds). On the other hand, the solver time for the MILP is in general higher than the one of our ILP. For example, for messages 23-cfg1, 24-cfg2 and 29-cfg3, we are unable to obtain a result in a reasonable amount of time using the MILP. In general, the worst-case complexity of our method is better than the method of [12] because the number of variables and constraints is fixed for a given problem instance, while in the case of [12] it increases in each iteration with no convergence guarantee and as a result can lead to memory overflow.

**Remark 3** : In [12], the problem of computing $X_m$ is show to be equivalent to a bin covering problem that is NP-hard. The heuristic schemes proposed in [12], [18] as well as the network calculus approach of [10], [11] sacrifice precision. Our solution of the problem is not only precise, but also improves the scalability compared to [12] as we have experimentally demonstrated through the benchmark application. Recalling that the computation of $R_m$ requires the computation of $X_m$ and $W_m$, where it turns out that the computation time of $W_m$ dominates, and so we can further improve the scalability of our approach by settling for a near optimal solution where $X_m$ is computed precisely as above, while $W_m$ is simply upper bounded by the value $T_{ST} + T_{fast}^m \times T_{HS}^m$ (since $m$ must be transmitted before this upper bound in its final cycle). Also note that the precise computation of $X_m$ in itself is benefitted by the existing heuristics that are being used to upper bound its solution space prior to exploring it.

### Table III

<table>
<thead>
<tr>
<th>Message N°</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td>cfg1</td>
<td>cfg2</td>
<td>cfg3</td>
<td>cfg1</td>
</tr>
<tr>
<td>Solver time (s)</td>
<td>33.59</td>
<td>11.65</td>
<td>1.22</td>
<td>29</td>
</tr>
<tr>
<td>$R_m$ (MT)</td>
<td>ILP</td>
<td>MILP</td>
<td>PPEPA</td>
<td>-</td>
</tr>
<tr>
<td>$R_m$ (MT)</td>
<td>ILP or MILP</td>
<td>PPEPA</td>
<td>-</td>
<td>365</td>
</tr>
<tr>
<td>Message N°</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>Configuration</td>
<td>cfg1</td>
<td>cfg2</td>
<td>cfg3</td>
<td>cfg1</td>
</tr>
<tr>
<td>Solver time (s)</td>
<td>3</td>
<td>240</td>
<td>2</td>
<td>23</td>
</tr>
<tr>
<td>$R_m$ (MT)</td>
<td>ILP</td>
<td>MILP</td>
<td>[12]</td>
<td>15</td>
</tr>
<tr>
<td>$R_m$ (MT)</td>
<td>ILP or MILP</td>
<td>N.S.</td>
<td>-</td>
<td>902</td>
</tr>
<tr>
<td>Message N°</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>Configuration</td>
<td>cfg1</td>
<td>cfg2</td>
<td>cfg3</td>
<td>cfg1</td>
</tr>
<tr>
<td>Solver time (s)</td>
<td>3.2</td>
<td>4</td>
<td>10.3</td>
<td>5</td>
</tr>
<tr>
<td>$R_m$ (MT)</td>
<td>ILP</td>
<td>MILP</td>
<td>[12]</td>
<td>22</td>
</tr>
<tr>
<td>$R_m$ (MT)</td>
<td>ILP or MILP</td>
<td>N.S.</td>
<td>N.S.</td>
<td>3.4</td>
</tr>
</tbody>
</table>

with content N.S. are those for which the solver stop after detecting that the message is unschedulable.

For each of the three configurations, we were able to compute using our method in a reasonable time for each message whether it is schedulable and in that case its precise worst-case response time. For $cfg1$ (resp. $cfg2$), messages 22 and 24-31 (resp. 26 and 28-31) are unschedulable, while no message misses its deadline under $cfg3$. With the method of [12], we are unable to compute in a reasonable time (within one hour) the response time of messages 20, 22, and 24-31 under $cfg1$ (the program exits after iteration 2 with error message "too much memory used" for messages 28-31); and of messages 22, 26 and 28-31 under $cfg2$. Moreover, under $cfg1$, the intermediate result of the response time of message 20 that we obtained (8 cycles filled in 3rd iteration in contrast to only 4 cycles filled for the precise value computation) shows that this message is deemed unschedulable while the precise worst-case response time obtained using our method shows the contrary. The same observation holds for message 23 under $cfg1$. Further the method of [12] yields a higher worst-case response time than the precise value computed by our method for messages 25 and 27 under $cfg2$.

Under $cfg3$, only one instance of every message interferes with the response time of lower priority messages, and as a result, the method of [12] terminates in a single iteration. In such case, both methods yield the same response time. In some such cases, the solver time of our method is higher owing to the inclusion of the minimal inter-arrival time constraints (12)-(13) and the corresponding variables, which happen to be redundant under the setting of $cfg3$. Indeed if we drop (12)-(13) from our ILP and add a constraint enforcing that only one instance of every message can be transmitted ($\sum_{j=1}^{U} x_j \leq 1$, $\forall i$), we obtain similar or better solver time than the method of [12] (for example, the solver time for computing the response time of message 31 in $cfg3$ with the above mentioned modification reduces from 8.75 seconds to 3.77 seconds). On the other hand, the solver time for the MILP is in general higher than the one of our ILP. For example, for messages 23-cfg1, 24-cfg2 and 29-cfg3, we are unable to obtain a result in a reasonable amount of time using the MILP. In general, the worst-case complexity of our method is better than the method of [12] because the number of variables and constraints is fixed for a given problem instance, while in the case of [12] it increases in each iteration with no convergence guarantee and as a result can lead to memory overflow.
V. Conclusion

We proposed an analytical method for determining the worst-case response time of FlexRay dynamic messages. We showed that the previous analysis method proposed in [12] is conservative because the proposed formulation doesn’t adequately consider the minimum interarrival time of messages. This previous method could then lead to making conservative decisions about the schedulability of dynamic messages, deciding certain schedulable messages as unschedulable. We presented a new formulation for computing the precise worst-case response time of FlexRay dynamic messages. Moreover in contrast to [12], which requires solving iteratively a sequence of ILPs, our method is non-iterative requiring solving two ILPs and hence computationally efficient. Our approach was validated by applying it to a SAE benchmark, and also compared against the approach of [12].

REFERENCES


