Safe Reconfiguration of LET Communication Intervals to Reduce End-to-End Latencies

Luiz Maia, Mario Günzel, and Gerhard Fohler
TU Dortmund University, Dortmund, Germany
University of Kaiserslautern-Landau, Germany

Safe Reconfiguration of LET Communication Intervals to Reduce End-to-End Latencies

Luiz Maia[1], Mario Günzel[2], and Gerhard Fohler[1]

[1] University of Kaiserslautern-Landau, Germany
{maianeto,fohler}@rptu.de
[2] TU Dortmund, Germany
mario.guenzel@tu-dortmund.de

Abstract. The Logical Execution Time (LET) model is a communication paradigm that ensures deterministic timing behavior by having fixed communication intervals, even for complex multi-rate cause-effect chains in multi-core systems. While the LET paradigm is widely used, it can introduce significant pessimism in form of longer end-to-end latencies if the communication intervals are not well configured. In the literature, approaches to optimize the end-to-end latency by reducing the communication intervals have been sparsely studied. In this work, we propose a novel method to optimize the end-to-end latency of multi-rate cause effect chains. To that end, we first exploit harmonic properties of the task set to apply phases to the read and write functions present in LET. Afterwards, we further reconfigure the communication intervals by adding precedence constraints. We prove that our measures are safe in the sense that they cannot increase the end-to-end latency, even in rare cases. Evaluation on a well established automotive benchmark as well as synthetic task sets show that our optimization method can reduce the end-to-end latencies of multi-rate cause-effect chains applying the LET paradigm.

1 Introduction

Modern *cyber-physical* systems (which can be found in medical technology, airplanes, cars, etc.) heavily rely on communication between sensors and actuators. A typical example is an application that consists of a task that reads the sensor (*cause*), a task that processes the read value, and a task that writes to an actuator (*effect*). This chained sequence of tasks executing and propagating data is known as a *cause-effect chain*. Depending on the latency variations encountered during the communication, the control performance of running applications can be impacted. For them, the *end-to-end latencies* between sender and receiving systems have to be within a given range, or the application's control performance suffers degradation leading to possible instability in the system [19]. As shown in the literature [10, 26], tasks present in a cause-effect chain can have different periods (multi-rate) and can be allocated to different cores resulting in parallel execution, which makes the end-to-end analysis non-trivial.

The Logical Execution Time (LET) model [11] reduce the complexity of analyzing the temporal requirements of multi-rate cause-effect chains by enabling timing and data flow determinism. This is achieved by introducing fixed communication points, i.e., communication only takes place at the beginning and at the end of *communication intervals*. While traditionally each communication interval spans the interval of two consecutive releases, the correct size and positioning of communication intervals is a key factor when optimizing the end-to-end latencies under LET. The literature can be divided into two different approaches: 1) optimizations to **minimize the end-to-end latency of specific cause-effect chains**, and 2) safely **shrinking the communication intervals** (i.e., making them subsets of the original intervals). While 1) usually achieves tight results, this potentially comes at the cost of increased end-to-end latency for other cause-effect chains in the system. This is a major limitation because a classical system can consist of up to 60 cause-effect chains [14]. On the other hand, 2) can be used to optimize the end-to-end latency of *all* cause-effect chains simultaneously. Therefore, 2) is suitable for the optimization of larger systems with several cause-effect chains, for systems where cause-effect chains are not even specified yet, and also prepares for potential changes of cause-effect chains during runtime.

In literature, different optimization approaches for 1) have been proposed, with the prominent ones being the approaches from Martinez et al. [20], Wang et al. [27], and Günzel et al. [7]. Specifically, [20] and [7] introduce task phases to optimize the position of communication intervals with respect to end-to-end latency, without changing their length. Wang et al. [27] optimize several parameters (such as end-to-end latency, time disparity, and jitter) by exploring both the length and position of the communication intervals.[3]

For 2), only two works have addressed the safe shrinking of communication intervals. Bradatsch et al. [5] were the first to propose a method that shrinks the right end of the communication intervals based on the response time of the tasks. Later, Maia et al. [16] went a step further and showed that by shrinking both ends of the communication intervals, end-to-end latencies of all chains can be even further optimized. In order to ensure that the shrinking is safe, i.e., that no job executes outside its communication interval, both approaches ensure that the underlying schedule where all tasks execute their *worst-case execution time* (WCET) is preserved. However, this conservative approach leaves the potential for optimization that actively modifies the schedule to reconfigure communication intervals.

Contribution: In this paper, we reduce the end-to-end latencies of multi-rate cause-effect chains by safely shrinking the communication intervals of LET. To that end, our approach expands on the limitations of [5, 16] by actively influencing the schedule to achieve better end-to-end latencies. Specifically, our contributions are as follows:

[3] Although [27] also applies an approach to shrink communication intervals (namely [16]) in a second step, their optimization does not achieve a safe shrinking in the sense of 2), because in the first step, communication intervals are shifted.

– We propose an optimization method that reconfigures communication intervals by applying task phases, which are determined by exploiting harmonic properties of cause-effect chains.
– We explore the optimization through the addition of precedence constraints.

In addition to these contributions, related work is discussed in Section 2, system model is introduced in Section 3, and existing approaches for shrinking communication intervals are detailed in Section 4.

2 Related Work

Assigning phases to tasks is a well known technique to increase schedulability, reduce the *worst-case response time* (WCRT) of tasks and their output jitter. Tindell [25] was the first to propose a method that applies static phases to tasks and showed how to perform the exact response time analysis of those tasks. In [22], Palencia and Harbour extended Tindell's work by considering tasks with static and dynamic phases in distributed systems. By considering a more accurate modeling of inter-task interference, Mäki-Turja and Nolin [18] proposed a tighter analysis for the assignment of phases to tasks as previous approaches calculate unnecessarily pessimistic response-time. In the avionics domain, Palencia et al. [21] presented a response time analysis that focus on hierarchically-schedulled time-partioned distributed systems. Although meaningful contributions to the problem of assigning phases to tasks, only Martinez et al. [20] looked to the problem from the perspective of cause-effect chains and their end-to-end latencies (see below).

When analyzing a multi-rate cause-effect chain in the automotive domain, the two most commonly considered latency metrics are: *reaction time* (*First to First* semantic) and *data age* (*Last to Last* semantic) [6]. The reaction time measures the *reactivity* of the system. It is the time interval between the occurrence of an external event until the *first* output based on that event. Data age measures the *freshness* of data. It is the time interval between a data sampled (read) by the first task in the cause-effect chain until the *last* output (actuation) based on such data is produced by the last task in the cause-effect chain. In [9], Günzel et al. showed that the values for the maximum reaction time (MRT) and maximum data age (MDA) are equivalent for multi-rate cause-effect chains.

Presented as part of the Giotto programming language, the LET paradigm [11] was initially introduced in the context of time-triggered tasks. Due to its timing and data-flow deterministic characteristics, the LET paradigm gained attention of the automotive industry in recent years. In [4], Biondi et al. showed how to realize the logical behavior of the LET paradigm by using additional dedicated tasks. An implementation on actual multi-core platforms for automotive systems was later presented in [3]. As a way to control accesses to shared memory and optimize the functional deployment on multi-core platforms, Pazzaglia et al. [23] used the LET paradigm to enforce causality and determinism.

Different techniques have been proposed to compute the end-to-end latencies of multi-rate cause-effect chains applying different communication paradigms (implicit communication, LET). In [2], Becker et al. used job-level dependencies to control data propagation on chains applying the implicit communication model. In [12], Klaus et al. proposed one way to enforce the execution behavior of precedence constraints during runtime. Based on a task dependency graph and assuming the LET paradigm, Kordon and Tang [13] proposed a method to determine the maximum data age. In [20], Martinez et al. presented a phase-aware LET analysis to improve the end-to-end latencies of specific multi-rate cause-effect chains. Bradatsch et al. proposed in [5] a method to reduce data age by setting the communication intervals equal to tasks' worst-case response time. Exploiting information from a feasible schedule, Maia et al. [16] proposed a method that safely shrinks and shifts the communication intervals of tasks applying the LET model. In [27], Wang et al. presented a two-step optimization method that also manipulates the length and position of the communication intervals. Their method does not achieve safe shrinking of all chains, because during the first optimization step, the communication intervals are shifted as in [20] and later optimized. In [17], Maia et al. proposed a method to reduce utilization by modifying the original task set. It uses precedence constraints to skip the execution of jobs which would work on already propagated data. Authors show that the end-to-end latencies are reduced for the now modified task set. Recently, Günzel et al. proposed in [7] an optimal phasing method to improve the end-to-end latencies of cause-effect chains applying the LET paradigm. Their method does not consider that the communication interval of the LET tasks can be less than their period interval.

We propose a method to safely reconfigure the communication interval of the LET paradigm beyond the intervals obtained by previous methods and without modifying the task set. We explore in detail the idea of applying phases to tasks in a way that the end-to-end latencies of all cause-effect chains are optimized instead of just one as proposed by Martinez et al. [20]. Moreover, we show that our phasing method can achieve equal or better latencies than Maia et al. [16] with much lower complexity and without the need of analyzing the full hyperperiod.

3 System Model

We consider a multi-core system composed of identical cores and a task set Γ containing periodic real-time tasks.

3.1 Tasks and Jobs

A task $\tau \in \Gamma$ is a tuple $(C_\tau, T_\tau, D_\tau, \phi_\tau)$, where $C_\tau > 0$ represents the *worst-case execution time* (WCET), T_τ is the period, D_τ is the deadline, and ϕ_τ is the phase. We assume tasks have constrained deadlines, i.e., the deadline is equal or less to the period. Each task τ releases jobs $\tau(i)$, $i \in \mathbb{N}^+$ recurrently, and we denote by J a generic job of a task τ. The job $\tau(i)$ is released at time $r_{\tau(i)} = \phi_\tau + (i-1)T_\tau$ and has an absolute deadline D_τ time units later. A schedule $\mathcal{S}$ specifies the execution behavior of all jobs of τ according to a given scheduling policy. The *start time* of $\tau(i)$ according to $\mathcal{S}$ is $s^{\mathcal{S}}_{\tau(i)}$, while the *finishing time* is $f^{\mathcal{S}}_{\tau(i)}$. If the referenced schedule $\mathcal{S}$ is clear, we omit the index $\mathcal{S}$ for brevity.

We assume that the tasks are scheduled using a preemptive task-level fixed-priority scheduler. That is, each task $\tau \in \Gamma$ has a given (unique) priority level π_τ, with $\pi_\tau < \pi_{\tau'}$ describing that τ has lower priority than τ'. At each time, among all *pending* jobs (i.e., released but not finished), the job of the task with the highest priority is executed. We assume that all jobs finish until their absolute deadline, which can be guaranteed by determining the *worst-case response time* (WCRT) $R_\tau := \sup_{\mathcal{S},i} f^{\mathcal{S}}_{\tau(i)} - r_{\tau(i)}$ using time-demand analysis [1] and checking if $R_\tau \leq D_\tau$.

As shown in [8, Propositions 13 and 14], the minimal start time is achieved if all jobs execute for 0 time units, and the maximal finishing times is achieved if all jobs execute for the WCET. Furthermore, if the execution time is fixed (to the WCET or to 0), then the schedule repeats after $\Phi(\Gamma) + 2H(\Gamma)$, with $\Phi(\Gamma) = \max(\{\phi_\tau \,|\, \tau \in \Gamma\})$ being the maximal phase and $H(\Gamma) = LCM(\{T_\tau \,|\, \tau \in \Gamma\})$ being the hyperperiod, as shown in [8, Lemma 16].

3.2 Communication Model

We assume communication under the LET paradigm, and model the communication through shared resources. Each job $\tau(i)$ has a *communication interval*

$$L_{\tau(i)} = [read_{\tau(i)}, write_{\tau(i)}], \tag{1}$$

where $read_{\tau(i)}$ and $write_{\tau(i)}$ specify when each $\tau(i)$ logically receives (reads) and logically transmits (writes) data within $L_{\tau(i)}$, respectively. The logical read and write operations occur periodically. That is, each task is equipped with a read-phase $\phi^R_\tau \in \mathbb{R}$ and a write-phase $\phi^W_\tau \in \mathbb{R}$, and the boundaries of the communication interval are derived as:

$$read_{\tau(i)} = (i-1)T_\tau + \phi^R_\tau, \qquad write_{\tau(i)} = (i-1)T_\tau + \phi^W_\tau \tag{2}$$

To ensure that no job starts before the data is read or finishes after the data is written, traditionally, the read-phase is set to the task phase $\phi^R_\tau := \phi_\tau$ and the write-phase is $\phi^W_\tau := \phi_\tau + D_\tau$. Figure 1 summarizes the notation of our communication model. As depicted, in the traditional model the read and write of each job only occurs at the boundaries of $L_{\tau(i)}$.

3.3 Cause-Effect Chain

A cause-effect chain represents an ordered sequence of communications carried out between a finite set of tasks. We represent a cause-effect chain by

$$E = (\tau_1 \rightarrow \tau_2 \rightarrow \cdots \rightarrow \tau_{|E|}), \tag{3}$$

with $|E|$ being the number of tasks in E, and $\tau_i \neq \tau_j$ for all $i \neq j$. For a pair of consecutive tasks τ_i and τ_{i+1} in E, the '$\rightarrow$' operator indicates that τ_{i+1} acts as a consumer/reader task, while τ_i as a producer/writer task. We assume that a task can only be part of a single cause-effect chain and that E samples (acquires data) at every read of its first task τ_1, i.e., at $read_{\tau_1(i)}$, $i \in \mathbb{N}^+$.

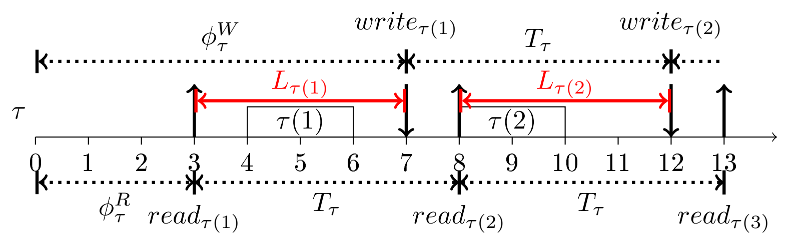


Fig. 1: Communication interval $L_{\tau(i)}$ for a given job $\tau(i)$ of task τ.

3.4 End-to-End Latency

Given a cause-effect chain, the end-to-end latency is the maximal time that it takes for data to traverse from one end of the chain to the other. When analyzing the latencies of multi-rate cause-effect chains, the most common metrics are: *Maximum reaction time (MRT):* How long does it take to fully process an external cause in the worst case? (This is also called the maximum button to action delay.) *Maximum data age (MDA):* How old is the data used in an actuation in the worst case? (This is also called the maximum data freshness.) As shown in [9], both MRT and MDA are equivalent. Therefore, we do not distinguish those metrics and use the term $E2E(E)$ universally to represent the *Maximum end-to-end latency* of a cause-effect chain E.

For the calculation of the end-to-end latency of multi-rate cause-effect chains, we follow the model from Günzel et al. [9]. Specifically, we start by determining the possible data propagation paths of E by using the notion of *job chains* c^E. That is, a job chain $c^E = (J_1 \to \cdots \to J_{|E|})$ is a sequence of jobs, such that the following requirements are respected:

(a) J_i is a job of τ_i, $i \in \{1, 2, \cdots, |E|\}$, i.e., $J_i = \tau_i(j_i)$ for some $j_i \in \mathbb{N}^+$.
(b) $write_{J_i} \leq read_{J_{i+1}}$ for all $i \in \{1, 2, \cdots, |E| - 1\}$.

To track the earliest propagation, *immediate forward* job chains fc_j^E with $j \in \mathbb{N}^+$ are considered, where $J_1 = \tau_1(j)$, and for all $i = 1, \ldots, |E|-1$ the job J_{i+1} is the earliest fulfilling (b). Furthermore, to track the origin of data, *immediate backward* job chains bc_j^E with $j \in \mathbb{N}^+$ are considered, where $J_{|E|} = \tau_{|E|}(j)$, and for all $i = |E| - 1, \ldots, 1$ the job J_i is the latest fulfilling (b).

During system's initial startup (*warm-up phase* [9]), immediate backward job chains cannot always be constructed for all $j \in \mathbb{N}^+$. That is, during the warm-up phase, job J_i might have no data to process because task τ_{i-1} has not been executed yet, i.e., the chain has not yet fully processed data for the first time. Therefore, we use $(\tau_1(W_1^E) \to \cdots \to \tau_{|E|}(W_{|E|}^E))$ to denote the first immediate backward job chain after the warm-up phase with $W_i^E \in \mathbb{N}$.

Since the LET paradigm enforces periodic communication intervals, they repeat every hyperperiod $H(E) = LCM(T_{\tau_1}, \cdots, T_{\tau_{|E|}})$ after the maximal phase $\Phi(E) = \max(\phi_{\tau_1}, \cdots, \phi_{\tau_{|E|}})$. Therefore, it is sufficient to calculate the end-to-end latencies until $\frac{H(E)}{T_{\tau_1}}$ jobs immediate forward job chains released after $\Phi(E)$. Specifically, this allows us to formulate

$$E2E(E) = T_1 + \max_{W_1^E < j \leq \xi} fc_j^E \tag{4}$$

with $\xi := \max(\left\lceil \frac{\Phi(E)-\phi_{\tau_1}}{T_{\tau_1}} \right\rceil, W_1^E) + \frac{H(E)}{T_{\tau_1}}$. Note that although our definition does not introduce ***augmented job chains*** as in [9] to define end-to-end latency, the definitions are still equivalent, since the sampling delay (i.e., the 'augmented' part) for MRT is exactly one period T_1 of τ_1 under the LET paradigm. Likewise, the additional actuation delay for MDA is exactly $T_{|E|}$ for $\tau_{|E|}$.

Table 1 summarizes the notations used throughout this paper.

Variable	Definition
$\tau \in \Gamma$	a task from task set Γ
$\tau(i), i \in \mathbb{N}^+$	i^{th} job of task τ
$read_{J_i}$	logical read-event of J_i
$write_{J_i}$	logical write-event of J_i
$H(E)$	hyperperiod of cause-effect chain E
$E(i)$	the i^{th} task in E, $i \in \{1, 2, \cdots, \|E\|\}$
J_i	a job of $E(i)$, $i \in \{1, 2, \cdots, \|E\|\}$
fc_j^E	immediate forward chain for the j^{th} job of τ_1 in E
bc_j^E	immediate backward chain for the j^{th} job of $\tau_{\|E\|}$ in E
R_τ	worst case response time of τ

Table 1: Notation Table

4 Shrinking Communication Intervals

In the LET paradigm, traditionally communication interval spans over the interval from release to deadline, which is achieved with a read-phase $\phi_\tau^R = \phi_\tau$ and a write-phase $\phi_\tau^W = \phi_\tau + D_\tau$. While this trivially ensures a safe communication interval, in the sense that for a feasible schedule every job always executes within its communication interval, those intervals can potentially be shrinked by *increasing the read-phase* or by *decreasing the write-phase*. Since a shrinked communication interval can speed up data propagation along all cause-effect chains in the system leading to reduced end-to-end latencies, there is a large gap for optimization in the classical LET paradigm. In the literature, to the best of our knowledge, there are two works that propose methods to safely shrink the communication intervals of the LET paradigm [5, 16].

Bradatsch et al. [5] were the first to propose a method to shrink the communication intervals. They consider synchronous tasks, i.e., $\phi_\tau = 0$, and shrink the communication intervals by decreasing the write-phase. That is, instead of the traditional setting for synchronous tasks, which is $\phi_\tau^R = 0$ and $\phi_\tau^W = D_\tau$, they decrease the write-phase to the WCRT $R_\tau = \sup_{\mathcal{S},i} f_{\tau(i)}^{\mathcal{S}} - r_{\tau(i)}$, i.e.,

$$\phi_\tau^R = 0 \qquad \text{and} \qquad \phi_\tau^W = R_\tau, \tag{5}$$

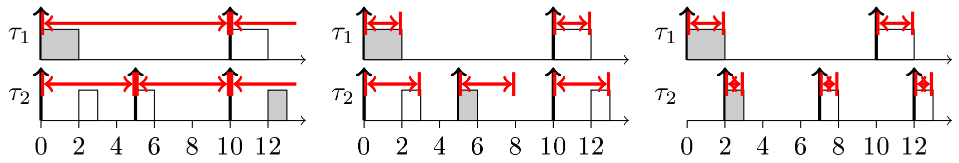
derived by classical WCRT analysis [1]. As a result, their communication intervals $L_{\tau(i)} = [(i-1)T_\tau, (i-1)T_\tau + R_\tau]$ are subsets of the traditional communication intervals for synchronous tasks $[(i-1)T_\tau, (i-1)T_\tau + D_\tau]$, which leads to significant improvement of end-to-end latencies.
Maia et al. [16] shrink the communication intervals from both sides and obtain better end-to-end latencies, without the limitation to synchronous task sets. In order to derive new safe bounds for the communication intervals, their method is based on the examination of the schedule $\mathcal{S}_{max}$, where each job executes for its full WCET. Specifically, instead of the traditional setting $\phi^R_\tau = \phi_\tau$ and $\phi^W_\tau = \phi_\tau + D_\tau$, they simulate the *earliest relative start* $ES^{max}_\tau = \inf_i s^{\mathcal{S}_{max}}_{\tau(i)} - r_{\tau(i)}$ and the *latest relative finish* $LF^{max}_\tau = \sup_i f^{\mathcal{S}_{max}}_{\tau(i)} - r_{\tau(i)}$ under the schedule $\mathcal{S}_{max}$, and set new read- and write-phases:

$$\phi^R_\tau = \phi_\tau + ES^{max}_\tau \qquad \text{and} \qquad \phi^W_\tau = \phi_\tau + LF^{max}_\tau, \qquad \forall \tau \in \Gamma \tag{6}$$

Maia et al. [16] show that, if the phase of the task is adjusted to $\phi_\tau := \phi^R_\tau$, the resulting communication intervals are safe even when jobs do not execute for their full WCET.
Both of these approaches are conservative in the sense that the modification of the task parameters has no impact on the underlying schedule $\mathcal{S}_{max}$ but only exploits the information on $\mathcal{S}_{max}$ to shrink the communication intervals. This leaves potential for further improvement by active modification of $\mathcal{S}_{max}$. However, careful treatment of the schedule modification is necessary to ensure that the task execution is not pushed outside its communication interval boundaries.

5 Exploiting Harmonic Properties

With respect to shrinking communication intervals, the current state of the art [16] shrinks the communication interval without modification of the schedule $\mathcal{S}_{max}$, as summarized in Section 4. This is achieved by ensuring that the phase ϕ_τ is not increased by more than ES^{max}_τ time units.
In order to showcase the potential of increasing the phase even further, we consider the example depicted in Figure 2, consisting of two tasks $\tau_1 = (2, 10, 10, 0)$ and $\tau_2 = (1, 5, 5, 0)$, with τ_1 having a higher priority than τ_2. Since the earliest relative start time is 0 for both tasks, the approach by Maia et al. [16] only reduces the write-offset, and leaves the read-offset and the phase at 0. However, applying a phase $\phi_{\tau_2} = 2$, we can reconfigure the communication interval even further. Please note that the data produced by job $\tau_1(1)$ is consumed by $\tau_2(3)$ with the traditional approach, by $\tau_2(2)$ with the approach by Maia et al. [16], and by $\tau_2(1)$ with our approach. Specifically, in this example, the immediate forward job chains have a length of 15 with the traditional configuration, of 8 with the configuration from [16], and of 3 with our approach. Therefore, the end-to-end latency of $E = (\tau_1 \rightarrow \tau_2)$ is reduced from $E2E(E) = 10 + 15 = 25$ to $E2E(E) = 10 + 8 = 18$ with the configuration from [16], and to $E2E(E) = 10 + 3 = 13$ with our approach.



(a) Traditional. (b) Maia et al. [16]. (c) Our.

Fig. 2: Different phasing strategies and communication intervals. Jobs belonging to the first immediate forward chain are marked in gray.

In this section, we identify cases where the phase can safely be enlarged by more than ES_τ^{max} while ensuring that the right end of the communication interval is not exceeded. We restrict our view to the synchronous case as a reference, where, by the critical instant theorem, the latest relative finish occurs with the first job and equals the WCRT R_τ. Therefore, the original write-operation $write_{\tau(i)}$ for job $\tau(i)$ occurs no earlier than at $(i-1)\cdot T_\tau + R_\tau$. Hence, our goal is to set phases $\phi_\tau > 0$ without enlarging the finish $f_{\tau(i)}^{\mathcal{S}_{max}}$ of any job $\tau(i)$ beyond $(i-1)\cdot T_\tau + R_\tau$. Note that although the work by Günzel et al. [7] also considers harmonic cause-effect chains, their method follows the conservative approach proposed by Martinez et al. [20], where the communication intervals of the LET paradigm are only shifted and remain at the length of tasks' period. Contrarily, in this work, we follow the approach by Maia et al. [16] to shrink communication intervals.
Enlarging the phases beyond the earliest start time modifies the schedule $\mathcal{S}_{max}$ and inevitably delays the finishing time of some jobs $\tau(i)$. To ensure that those jobs with increased finish do not exceed $(i-1)\cdot T_\tau + R_\tau$, we need to make sure that they do *not* experience *more* interference from a higher priority task than the job with the latest finish, which in the synchronous case is the first job. For example, if in Figure 2 τ_1 would have a period of 7, then the phase of τ_2 could not be increased without increasing additional interference for $\tau_2(2)$. As a result, $\tau_2(2)$ would suffer from larger interference than $\tau_2(1)$ and its execution would exceed the right end of the communication interval at time 8.
In general, as long as we can ensure that the interference of higher priority (hp) jobs $\tau' \in \mathrm{hp}(\tau)$ on subsequent jobs $\tau(i)$ (with $i > 1$) is upper bounded by the interference on $\tau(1)$ then we can safely increase the phase. The following Lemma shows that this is achieved in cases where the job releases follow a *harmonic* pattern, i.e., if $T_{\tau'}$ divides T_τ or T_τ divides $T_{\tau'}$ for all $\tau' \in \mathrm{hp}(\tau)$.

Lemma 1. *Let $\tau \in \Gamma$, such that for all tasks τ' that have higher priority than τ, i.e., for all $\tau' \in \mathrm{hp}(\tau)$, we have $\phi_{\tau'} = 0$ and*

$$T_{\tau'} \mid T_\tau \qquad or \qquad T_\tau \mid T_{\tau'}. \tag{7}$$

Then, any phase $\phi_\tau \in [0, \max_{\tau' \in \mathrm{hp}(\tau)} f_{\tau'(1)}^{\mathcal{S}_{max}}]$ ensures that the first job $\tau(1)$ has the latest relative finish, i.e., $f_{\tau(1)}^{\mathcal{S}_{max}} - r_{\tau(1)} \geq f_{\tau(i)}^{\mathcal{S}_{max}} - r_{\tau(i)}$ for all $i \in \mathbb{N}^+$.[4] *Furthermore, $f_{\tau(1)}^{\mathcal{S}_{max}} \leq R_\tau$, and therefore, $f_{\tau(i)}^{\mathcal{S}_{max}} \leq (i-1)\cdot T_\tau + R_\tau$ for all $i \in \mathbb{N}^+$.*

[4] For convenience of the presentation we define $\max_{\tau' \in \mathrm{hp}(\tau)} f_{\tau'(1)}^{\mathcal{S}_{max}} = 0$ if $\mathrm{hp}(\tau) = \emptyset$.

In the proof, we utilize formulas to compute $f_{\tau(i)}^{\mathcal{S}_{max}}$ based on a busy-interval. Formal proof for these formulas is provided in Appendix A (Lemmas 3 and 4).

Proof. First, we consider the case $\text{hp}(\tau) = \emptyset$. Then τ is the highest-priority task and has the phase $\phi_\tau = 0$. For the highest priority task, in $\mathcal{S}_{max}$, all jobs $\tau(i)$ have the same relative finish, namely $f_{\tau(i)}^{\mathcal{S}_{max}} - r_{\tau(i)} = C_\tau = R_\tau$, and $f_{\tau(i)}^{\mathcal{S}_{max}} = (i-1) \cdot T_\tau + R_\tau$. Therefore, the lemma holds for this case.
In the remainder of the proof, we assume $\text{hp}(\tau) \neq \emptyset$, i.e., τ is not the highest-priority task. We examine the relative finish of a job $\tau(i)$, namely $f_{\tau(i)}^{\mathcal{S}_{max}} - r_{\tau(i)}$. By Lemma 3, for all $t \in [r_{\tau(i)}, f_{\tau(i)}^{\mathcal{S}_{max}})$ we have

$$t < b + C_\tau + \sum_{\tau' \in \text{hp}(\tau)} \eta_{\tau'}(b,t) \cdot C_{\tau'}, \tag{8}$$

where $b \le r_{\tau(i)}$ is the latest time point such that higher priority tasks $\text{hp}(\tau)$ are executed during $[b, r_{\tau(i)}]$ in $\mathcal{S}_{max}$ and all jobs of $\text{hp}(\tau)$ released before b finish at or before b in $\mathcal{S}_{max}$, and where $\eta_{\tau'}(b,t) := \left\lceil \frac{\max(t-\phi_{\tau'},0)}{T_{\tau'}} \right\rceil - \left\lceil \frac{\max(b-\phi_{\tau'},0)}{T_{\tau'}} \right\rceil$ is the number of job releases of task τ' in the interval $[b,t)$.
First, we prove that $b \ge (i-1) \cdot T_\tau$. Specifically, this only requires to show that all jobs of $\tau' \in \text{hp}(\tau)$, which are released before $(i-1) \cdot T_\tau$, also finish until $(i-1) \cdot T_\tau$. We consider a job $\tau'(j)$, $j \in \mathbb{N}^+$ with

$$r_{\tau'(j)} = (j-1) \cdot T_{\tau'} < (i-1) \cdot T_\tau, \tag{9}$$

and show that $f_{\tau'(j)} \le (i-1) \cdot T_\tau$ by distinguishing two different cases from (7):

- **Case 1:** $T_{\tau'} \mid T_\tau$. Due to (9), we obtain $j \cdot T_{\tau'} \le (i-1) \cdot T_\tau$. Since the task set is schedulable by assumption of the system model, we exploit $R_{\tau'} \le D_{\tau'} \le T_{\tau'}$ to obtain $f_{\tau'(j)} \le r_{\tau'(j)} + R_{\tau'} = (j-1) \cdot T_{\tau'} + R_{\tau'} \le j \cdot T_{\tau'} \le (i-1) \cdot T_\tau$.
- **Case 2:** $T_\tau \mid T_{\tau'}$. Let $\xi \in \mathbb{N}^+$ such that $T_\tau \cdot \xi = T_{\tau'}$. Then, $(j-1) \cdot T_{\tau'} = (j-1) \cdot \xi \cdot T_\tau$, which is $< (i-1) \cdot T_\tau$ because of (9). Since τ' has higher priority than τ, we know $R_{\tau'} \le R_\tau$. Furthermore, since the task set is schedulable, $R_\tau \le D_\tau \le T_\tau$. We conclude $f_{\tau'(j)} \le r_{\tau'(j)} + R_{\tau'} = (j-1) \cdot \xi \cdot T_\tau + R_{\tau'} \le (j-1) \cdot \xi \cdot T_\tau + T_\tau$. This is $\le (i-1) \cdot T_\tau$ because $(j-1) \cdot \xi \cdot T_\tau < (i-1) \cdot T_\tau$.

In both cases we have shown that $f_{\tau'(j)} \le (i-1) \cdot T_\tau$ whenever (9) holds. This concludes the proof of $b \ge (i-1) \cdot T_\tau$.
Next, we consider the term $\eta_{\tau'}(b,t)$ from Equation (8). Intuitively, $\eta_{\tau'}(b,t)$ determines the impact that τ' has on the job $\tau(i)$. To show that the first job of τ suffers from at least the same interference as $\tau(i)$, in the following we prove

$$\eta_{\tau'}(b,t) \le \eta_{\tau'}(b - (i-1) \cdot T_\tau, t - (i-1) \cdot T_\tau). \tag{10}$$

Again, we do this by distinguishing the two different cases from Equation (7):

- **Case 1:** $T_{\tau'} \mid T_\tau$. For any job release $r_{\tau'(j)} \in [b,t)$, we know that there is a job release $r_{\tau'(j-(i-1)\frac{T_\tau}{T_{\tau'}})} \in [b - (i-1) \cdot T_\tau, t - (i-1) \cdot T_\tau)$. The job $\tau'(j - (i-1)\frac{T_\tau}{T_{\tau'}})$ exists, because

$$j \cdot T_{\tau'} > (j-1) \cdot T_{\tau'} \ge b \ge (i-1) \cdot T_\tau \tag{11}$$

– **Case 2:** $T_\tau \mid T_{\tau'}$. If there is no job release of τ' during $[b, t)$, then Equation (10) trivially holds. Otherwise, if there is a job release $r_{\tau'(j)} \in [b, t)$, then we use that $[b, t) \subseteq [(i-1) \cdot T_\tau, i \cdot T_\tau)$ to derive $r_{\tau'(j)} = (j-1) \cdot T_{\tau'} \in [(i-1) \cdot T_\tau, i \cdot T_\tau)$. Since $T_\tau \mid T_{\tau'}$, this means that

$$(j-1) \cdot T_{\tau'} = (i-1) \cdot T_\tau. \tag{12}$$

We conclude that for the job $\tau'(j)$, there is a corresponding job $\tau'(j - \frac{(i-1) \cdot T_\tau}{T_{\tau'}})$ with release in $[b - (i-1) \cdot T_\tau, t - (i-1) \cdot T_\tau)$.

With the two cases, we conclude that (10) holds. In the following, we apply Equation (10) to Equation (8) to conclude the lemma. Specifically, we obtain $t < b + C_\tau + \sum_{\tau' \in \text{hp}(\tau)} \eta_{\tau'}(b - (i-1) \cdot T_\tau, t - (i-1) \cdot T_\tau) \cdot C_{\tau'}$, and therefore

$$t - (i-1)T_\tau < b - (i-1)T_\tau + C_\tau + \sum_{\tau' \in \text{hp}(\tau)} \eta_{\tau'}(b - (i-1)T_\tau, t - (i-1)T_\tau) C_{\tau'} \tag{13}$$

for all $t \in [r_{\tau(i)}, f_{\tau(i)}^{\mathcal{S}_{max}})$. By the choice of $\phi_\tau \in [0, \max_{\tau' \in \text{hp}(\tau)} f_{\tau'(1)}^{\mathcal{S}_{max}}]$, we know that the system is busy executing jobs of $\text{hp}(\tau)$ in $[0, \phi_\tau]$. Furthermore, $0 \leq b - (i-1) \cdot T_\tau \leq r_{\tau(i)} - (i-1) \cdot T_\tau = \phi_\tau$. Therefore, we know that the system is busy executing jobs of $\text{hp}(\tau)$ in $[0, b - (i-1) \cdot T_\tau]$. Hence, $\sum_{\tau' \in \text{hp}(\tau)} \eta_{\tau'}(0, b - (i-1) \cdot T_\tau) C_{\tau'} \geq b - (i-1) \cdot T_\tau$, and by using this and the additivity of $\eta_{\tau'}$ in Equation (13), we obtain

$$t - (i-1) \cdot T_\tau < C_\tau + \sum_{\tau' \in \text{hp}(\tau)} \eta_{\tau'}(0, t - (i-1) \cdot T_\tau) \cdot C_{\tau'} \tag{14}$$

for all $t \in [r_{\tau(i)}, f_{\tau(i)}^{\mathcal{S}_{max}})$. This is equivalent to

$$t < C_\tau + \sum_{\tau' \in \text{hp}(\tau)} \eta_{\tau'}(0, t) \cdot C_{\tau'} = C_\tau + \sum_{\tau' \in \text{hp}(\tau)} \left\lceil \frac{t}{T_{\tau'}} \right\rceil \cdot C_{\tau'} \tag{15}$$

for all $t \in [r_{\tau(i)} - (i-1) \cdot T_\tau, f_{\tau(i)}^{\mathcal{S}_{max}} - (i-1) \cdot T_\tau) = [\phi_\tau, f_{\tau(i)}^{\mathcal{S}_{max}} - (i-1) \cdot T_\tau)$. Applying this to the calculation of $f_{\tau(1)}^{\mathcal{S}_{max}}$ from Lemma 4 from the appendix, we obtain that $f_{\tau(1)}^{\mathcal{S}_{max}} \geq f_{\tau(i)}^{\mathcal{S}_{max}} - (i-1) \cdot T_\tau$. Therefore, $f_{\tau(i)}^{\mathcal{S}_{max}} - r_{\tau(i)} = f_{\tau(i)}^{\mathcal{S}_{max}} - (i-1) \cdot T_\tau - \phi_\tau \leq f_{\tau(1)}^{\mathcal{S}_{max}} - \phi_\tau = f_{\tau(1)}^{\mathcal{S}_{max}} - r_{\tau(1)}$, i.e., $\tau(1)$ has the latest relative finish of all jobs of τ. Since the phase modification has no impact on the schedule for the first job $\tau(1)$, still $f_{\tau(1)}^{\mathcal{S}_{max}} \leq R_\tau$ holds. Specifically, we derive $f_{\tau(i)}^{\mathcal{S}_{max}} \leq (i-1) \cdot T_\tau + R_\tau$ for all $i \in \mathbb{N}^+$. This proves the lemma.

Referring back to the example in Figure 2, we have $T_{\tau_2} \mid T_{\tau_1}$. Therefore, by Lemma 1, we can enlarge the phase of τ_2 to $f_{\tau'(1)}^{\mathcal{S}_{max}} = 2$ while ensuring that all jobs finish within their communication interval, as depicted in the right image. If we increase the phase ϕ_τ of a task τ according to Lemma 1, then all jobs of lower priority tasks $\tilde{\tau} \in \text{lp}(\tau)$ still finish before their WCRT. However, when we want to do another phase modification for one of the lower priority tasks $\tilde{\tau} \in \text{lp}(\tau)$, Lemma 1 is not sufficient anymore, because there are some tasks in $\text{hp}(\tilde{\tau})$ (specifically, $\tau \in \text{hp}(\tilde{\tau})$) with phases involved. Therefore, to allow for phase modification of several tasks, we need to weaken the assumption '*for all* $\tau' \in \text{hp}(\tau)$*, we have* $\phi_{\tau'} = 0$' from Lemma 1, which we do in the following lemma.

Lemma 2. *Let Γ be a task set, such that for all tasks $\tau \in \Gamma$ we either have a phase of $\phi_\tau \in [0, \max_{\tau' \in \text{hp}(\tau)} f^{\mathcal{S}_{max}}_{\tau'(1)}]$ if*

$$T_{\tau'} \mid T_\tau \qquad \textit{or} \qquad T_\tau \mid T_{\tau'} \tag{16}$$

for all $\tau' \in \text{hp}(\tau)$, or a phase of $\phi_\tau = 0$ otherwise. Then, $f^{\mathcal{S}_{max}}_{\tau(i)} \leq (i-1) \cdot T_\tau + R_\tau$ holds for all tasks $\tau \in \Gamma$ and for all $i \in \mathbb{N}^+$. Furthermore, for tasks $\tau \in \Gamma$ for which Equation (16) *holds, the first job $\tau(1)$ has the latest relative finish.*

Again, for the proof, we rely on the formulas to derive $f^{\mathcal{S}_{max}}_{\tau(i)}$ provided in Appendix A. For the parts of the proof that coincide with the proof strategy from Lemma 1, we just explain how the proof is adjusted to be applicable here.

Proof. We prove this lemma by induction over the tasks in Γ ordered by priority, starting with the highest-priority task.
Base Case: Let $\tau \in \Gamma$ be the highest-priority task. Then, by assumption $\phi_\tau = 0$. Since $R_\tau = C_\tau$ for the highest-priority task, $f^{\mathcal{S}_{max}}_{\tau(i)} = r_{\tau(i)} + R_\tau = (i-1) \cdot T_\tau + R_\tau$, and all jobs have the same relative finish. This proves the base case.
Induction step: Let $\tau \in \Gamma$, and assume that $f^{\mathcal{S}_{max}}_{\tau'(i)}$ for all $\tau' \in \text{hp}(\tau)$ and $i \in \mathbb{N}^+$, and further $\tau'(1)$ has the latest relative finish of τ' if (16) holds for τ'. We distinguish two cases: First, if (16) does not hold for τ, then $\phi_\tau = 0$, and $f^{\mathcal{S}_{max}}_{\tau(i)} \leq r_{\tau(i)} + R_\tau = (i-1) \cdot T_\tau + R_\tau$ for all $i \in \mathbb{N}^+$. Second, if (16) holds for τ, then $\phi_\tau \neq 0$ is allowed. For this case, we follow the same arguments as in the proof of Lemma 1, but account for the more relaxed assumption on the phase of higher-priority tasks as follows:

- We can still apply Lemma 3, even under the more relaxed setting. Therefore, Equation (8) still holds.
- We obtain $b \geq (i-1) \cdot T_\tau$, because any job $\tau'(j)$ of a higher-priority task $\tau' \in \text{hp}(\tau)$ released before $(i-1) \cdot T_\tau$, i.e., $r_{\tau'(j)} < (i-1) \cdot T_\tau$, has a finish of $f^{\mathcal{S}_{max}}_{\tau'(j)} \leq (j-1) \cdot T_{\tau'} + R_{\tau'}$ by induction step. Using $R_{\tau'} \leq T_{\tau'}$ and $R_{\tau'} \leq R_\tau \leq T_\tau$, we can prove $b \geq (i-1) \cdot T_\tau$ using the same strategy as in the proof of Lemma 1.
- We still obtain $\eta_{\tau'}(b, t) \leq \eta_{\tau'}(b - (i-1) \cdot T_\tau, t - (i-1) \cdot T_\tau)$ from Equation (10) by following the same steps. The only two differences are that in Equation (11), we have $j \cdot T_{\tau'} > (j-1) \cdot T_{\tau'} + \phi_{\tau'} \geq b$, because $\phi_{\tau'} \leq \max_{\tau'' \in \text{hp}(\tau')} R_{\tau''} < R_{\tau'} \leq T_{\tau'}$, and Equation (12) is derived from the fact that $r_{\tau'(j)} = (j-1) \cdot T_{\tau'} + \phi_{\tau'} \in [(i-1) \cdot T_\tau, i \cdot T_\tau)$ with $\phi_{\tau'} \in [0, T_{\tau'})$.
- In Equation (15), $C_\tau + \sum_{\tau' \in \text{hp}(\tau)} \eta_{\tau'}(0, t) \cdot C_{\tau'}$ is not simplified to $C_\tau + \sum_{\tau' \in \text{hp}(\tau)} \left\lceil \frac{t}{T_{\tau'}} \right\rceil \cdot C_{\tau'}$ but to $C_\tau + \sum_{\tau' \in \text{hp}(\tau)} \left\lceil \frac{t - \phi_{\tau'}}{T_{\tau'}} \right\rceil \cdot C_{\tau'}$. However, this term can still be applied to Lemma 4 to prove the lemma in the same way.

By following the proof of Lemma 1 using the modifications described above, this concludes the proof of the induction step.

Based on this lemma, we can traverse the task set from highest-priority to lowest-priority task, and increase the phase to $\max_{\tau' \in \text{hp}(\tau)} f^{\mathcal{S}_{max}}_{\tau'(1)}$ when the harmonic properties from Equation (16) are fulfilled. Furthermore, as stated in Lemma 2, the latest relative finish of a task τ with modified phase can be computed by determining the finish of the first job only. As proven in Lemma 4 of Appendix A, this corresponds to finding the earliest $t \geq \phi_\tau$ such that:

$$t \geq C_\tau + \sum_{\tau' \in \text{hp}(\tau)} \left\lceil \frac{t - \phi_{\tau'}}{T_{\tau'}} \right\rceil \cdot C_{\tau'} \tag{17}$$

Specifically, the latest relative finish can be computed using fixed-point iteration starting with $t = \phi_\tau$ and updating t using Equation (17). Our proposed phasing and communication offsets are summarized in Algorithm 1.

Algorithm 1 Proposed phasing and communication offsets.

```
1: Sort $\Gamma$ by priority (highest priority first).
2: for $\tau \in \Gamma$ do
3:     if $T_{\tau'} \mid T_\tau$ or $T_\tau \mid T_{\tau'}$ for all $\tau' \in \text{hp}(\tau)$ then
4:         $\phi_\tau := \max_{\tau' \in \text{hp}(\tau)} f^{\mathcal{S}_{max}}_{\tau'(1)}$        ▷ $f^{\mathcal{S}_{max}}_{\tau'(1)}$ calculated using Equation (17).
5:         Calculate $f^{\mathcal{S}_{max}}_{\tau(1)}$ with phasing $\phi_\tau$.        ▷ Using Equation (17).
6:         $\phi^R_\tau := \phi_\tau$ and $\phi^W_\tau := f^{\mathcal{S}_{max}}_{\tau(1)}$
7:     else
8:         $\phi_\tau := 0$, $\phi^R_\tau := 0$, and $\phi^W_\tau := R_\tau$.
9:     end if
10: end for
```

This phasing and the ones applied to the read and write operations ensure that the communication intervals are safe (in the sense that no job is executed outside its communication interval), and it dominates the traditional phasing as well as the shrinked communication intervals from Bradatsch et al. [5]. This is proven in the following theorem.

Theorem 1. *The proposed phasing and communication offsets provided in Algorithm 1 are safe in the sense that*

$$f^{\mathcal{S}}_{\tau(i)} \leq (i-1) \cdot T_\tau + \phi^W_\tau \tag{18}$$

for all tasks $\tau \in \Gamma$ and for all schedules $\mathcal{S}$. Furthermore, $\phi^W_\tau \leq R_\tau$, i.e., the approach dominates the approach from Bradatsch et al. [5].

Proof. Consider Γ with the phasing obtained from Algorithm 1. Let $\mathcal{S}$ be a schedule of the task set. First, the latest finishing time among all schedules can be observed when executing all jobs for their WCET, i.e., $f^{\mathcal{S}}_{\tau(i)} \leq f^{\mathcal{S}_{max}}_{\tau(i)}$ (as proven for example in [8, Proposition 13]). The phasing of tasks in Γ suffices the conditions of Lemma 2. Therefore, the latest relative finish is obtained for

$\tau(1)$. Specifically, this means that $f^{\mathcal{S}_{max}}_{\tau(i)} - r_{\tau(i)} \leq f^{\mathcal{S}_{max}}_{\tau(1)} - r_{\tau(1)}$. We conclude $f^{\mathcal{S}_{max}}_{\tau(i)} \leq f^{\mathcal{S}_{max}}_{\tau(1)} + r_{\tau(i)} - r_{\tau(1)} = f^{\mathcal{S}_{max}}_{\tau(1)} + (i-1) \cdot T_\tau = \phi^W_\tau + (i-1) \cdot T_\tau$. This proves Equation (18).
Furthermore, by Lemma 2, $f^{\mathcal{S}_{max}}_{\tau(1)} \leq R_\tau$, and therefore $\phi^W_\tau \leq R_\tau$. Since further $\phi^R_\tau = 0$, this proves that communication intervals under our approach are subsets of the communication intervals derived by Bradatsch et al. [5]. Therefore, our approach dominates [5].

While the communication intervals proposed in Algorithm 1 dominate the communication intervals from Bradatsch et al. [5] due to Theorem 1, our solution does not dominate the approach by Maia et al. [16]. The reason is that Algorithm 1 is more concerned with phase modifications that alter the schedule $\mathcal{S}_{max}$ and leaves some potential improvements for the tasks that fall into the 'else'-case on line 7. In Section 6, we propose a method that takes care of those tasks and optimize the phases of their read and write operations.

6 Exploring Interval Configuration Through Precedence Constraints

In this section, we show how precedence constraints, i.e., restrictions that delay the start of a job's execution, can be combined with our phasing method to safely reconfigure communication intervals. By interval reconfiguration we mean finding the read and write offset combination that simultaneously best reduces the end-to-end latencies of the cause-effect chains in the system. In Figure 3, we show that compared to models that focus solely on shrinking the communication intervals of all tasks, the combination of phasing and precedence constraints results in reduced end-to-end latencies.

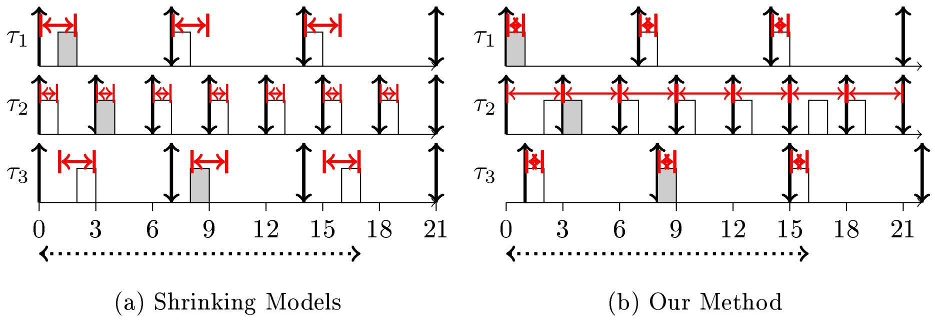


Fig. 3: End to end latency difference between methods that shrink the intervals of all tasks and our reconfiguration method.

In Figure 3, we show that by establishing two precedence constraints (see below), our method obtained interval configurations that reduced the end-to-end latency of $E = (\tau_1 \to \tau_2 \to \tau_3)$ from $E2E(E) = 7 + 10 = 17$ to $E2E(E) = 7 + 9 = 16$.

The first established precedence is between the first jobs of τ_2 and τ_3, i.e., $\tau_3(1) \prec \tau_2(1)$. The second precedence is between the third job of τ_3 and the sixth job of τ_2, i.e., $\tau_3(3) \prec \tau_2(6)$. Note that once a precedence constraint is established, it repeats in every hyperperiod. In Figure 3, we marked in gray the jobs belonging to the first immediate forward chain.
As explained in Section 4, the parameters used by [16] to define the boundaries of communication intervals are the earliest relative start ($ES_\tau^{\max}$) and latest relative finish ($LF_\tau^{\max}$) time of a task. By establishing precedence constraints between specific jobs from different tasks, our method controls the $ES_\tau^{\max}$ and $LF_\tau^{\max}$ values of those tasks, which in turn allows our method to control their communication intervals. Note that depending on which precedence constraints are established by our method, different interval configurations can be defined as each precedence constraint affects $ES_\tau^{\max}$ and $LF_\tau^{\max}$ values differently. The methods in [2, 12] can be used to enforce precedence constraints during runtime.

6.1 Tree Search

We model our problem of reconfiguring communication intervals by means of phasing and precedence constraints as a level order tree search. ILP or SMT solvers are also possible options but for convenience we used the framework in [24] to generate our tree. Figure 4 shows the structure of the tree and how we modeled its nodes and edges.

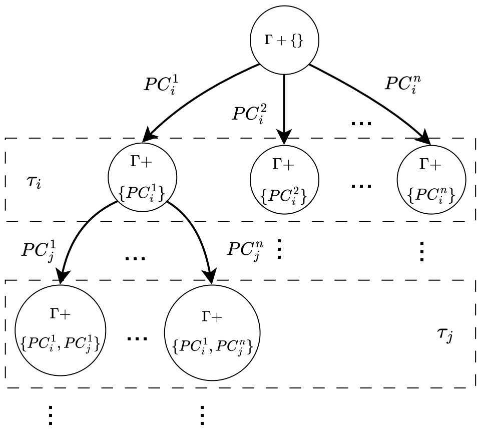


Fig. 4: Structure of search tree

The root node represents a schedulable task set Γ according to a fixed-priority scheduling policy. Note that the set of precedence constraints is empty at the root node. An edge (PC_i^n) represents a single precedence constraint between two jobs from different tasks in Γ, where PC_i^n is the n^{th} precedence established to a job of task τ_i. Each node represents task set Γ plus the set of precedence constraints established at each edge on the path between the root node and the current node. At each level of the tree, jobs of a given task in Γ establish precedence relations with jobs from other tasks in Γ (see Section 6.2). Note that multiple jobs of a

task can establish multiple precedence relations. Therefore, a task could be on more than one level of the tree. A node is considered a solution node when the set of precedence constraints established so far result in communication intervals that lead to reduced end-to-end latencies of all chains present in the system.
For each node in the tree, our method computes the communication intervals of each task based on the technique shown in [16] and recomputes the end-to-end latencies of all cause-effect chains. If the set of precedence constraints present in a node resulted in larger end-to-end latencies for one or more chains, our method assigns penalty points based on the number of chains that were affected. Likewise, if the set of precedence constraints resulted in shortened end-to-end latencies, our method assigns bonus points to the node. After sorting the nodes based on their points and compliance with our heuristic function (see Section 6.2), our method selects which precedence constraint should be added to the task set under consideration. If all precedence constraints worsened latency values, our method backtracks to the node at the previous level of the tree and selects a new node to investigate. Below we describe how our method generates nodes and how it decides which path to follow in the tree.

6.2 Heuristic Function

When establishing and evaluating precedence constraints for a task τ, our method considers three aspects: (i) τ's position in the chain, (ii) the period relation between τ, its predecessor and successor in the chain, and (iii) which jobs of τ define the boundaries of its communication intervals. We say a job J defines the interval boundaries of τ if $s_J^{S_{max}} - r_J = ES_\tau^{\max}$ or $f_J^{S_{max}} - r_J = LF_\tau^{\max}$. For each job J of τ that fulfills (iii), our method creates a set γ containing jobs from other tasks which can be scheduled within the release and deadline of J. Each job in γ is a *candidate* to establish a precedence with J. That is, for each J fulfilling (iii) and for each *candidate* in γ, our method generates a child node if a feasible schedule is obtained after establishing the precedence constraint, i.e., adding *candidate* $\prec J$ to Γ results in a schedulable task set.
In order to perform node evaluation, our heuristic considers that a cause-effect chain consists of three segments: *head*, *body* and *tail*. Depending on which segment of the chain a task τ belongs to, our heuristic uses different strategies to evaluate the effectiveness of a given precedence constraint. The number of tasks in the chain defines which and how many tasks are part of each segment.
We say a task is part of the *head* segment if it is located among first third of tasks present in the chain. If a task is on the second third, it belongs to the *body* segment, otherwise, it is part of the *tail* segment.[5] Below, we detail how our heuristic expects the communication intervals of τ to be configured depending on which segment the chain task τ is part of.

[5] If the number of tasks in the chain is a multiple of three, each segment has the same length. If not, the head and tail segments have their length equal to the quotient, while the body segment has its length equal to the quotient plus the remainder.

Tail Segment: For a task τ in the *tail* segment, our heuristic favors precedence constraints that: (i) result in intervals that are short, (ii) minimize the maximum time interval between the read offset of τ and the write offset of its predecessor in the chain. The rationale is, by having short intervals in the tail segment, our heuristic reduces the output delay of τ, which contributes to reduce the overall end-to-end latencies of the chain. Likewise, minimizing the maximum time interval between the read offset of τ and the write offset of its predecessor reduces the data propagation delay.

Head Segment: For a task τ in the head segment, if τ is the first task in the chain, our method favors precedence constraints that result in intervals that start close to the beginning of τ's period interval. The rationale is, by having intervals positioned close to the beginning of τ's period interval, we reduce the amount of phasing that might have to be added for all the subsequent tasks in the chain. If τ is not the first task in the chain, our method favors the reconfiguration of τ as done in the tail segment.

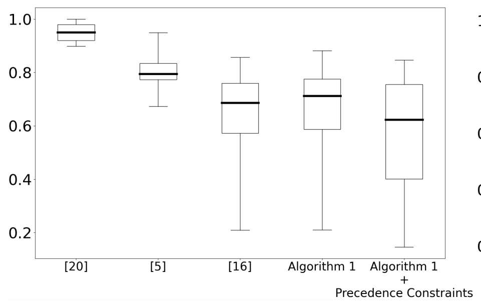
Body Segment: For a task τ positioned in the body segment, our method reconfigures the intervals of τ depending on the period relation between τ, its predecessor and its successor in the chain. More specifically, if task τ is part of an under(over)-sampling relation with its predecessor and part of an over(under)-sampling relation with its successor, then our method favors the non reconfiguration of τ's intervals. The rationale is that, since the successor or predecessor of τ cannot keep up with its sampling rate, reducing the communication intervals of τ significantly will not necessarily reduce data propagation delay. In this case, it is more beneficial to keep τ's communication interval longer and allow precedence constraints to shape the intervals of its successor or predecessor, so that they align better with the intervals of τ. In this case, our method generates a node that contains the same set of precedence constraints as the parent node and assign more bonus points to it. Note that our method also generates nodes based on possible precedence constraints to τ. However, those nodes do not receive bonus points which make them less likely to be chosen as the next node to be investigated. For all the other scenarios, our method favors the reconfiguration of τ as done in the tail segment.

7 Experimental Results

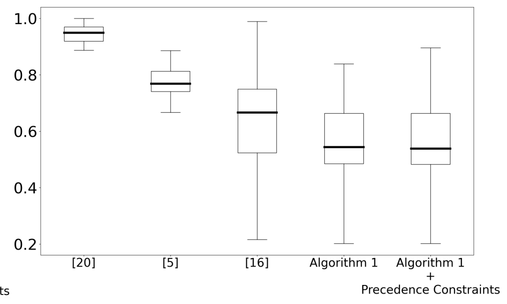
We evaluate our work based on the Real World Automotive Benchmarks presented by BOSCH [14] and synthetic task sets. We compare our method with the approaches presented by Martinez et al. [20], Bradatsch et al. [5] and Maia et at. [16]. As done by Martinez et al. [20], when evaluating their method, we considered that only the last two tasks in the cause-effect chain could receive an offset. For all the methods, we consider that the task sets run on a system comprised of four cores, are allocated according to the worst-fit strategy, and are scheduled according to fixed-priority. We used the method proposed by Maia et at. [16] as a baseline for our heuristic function. For this set of experiments, we allowed our method to search the tree for 5 minutes per task set.

7.1 Real World Automotive Benchmarks

We generated and tested 700 schedulable task sets based on the parameters of the Real World Automotive Benchmarks [14]. We assign periods to tasks following the definitions of Table III in [14]. The range of possible periods is: $\{1, 2, 5, 10, 20, 50, 100, 200, 1000\}$ms. Note that the sum of the probabilities for possible periods in [14] is 85%. The remaining 15% is for angle-asynchronous tasks. Since, we do not consider angle-asynchronous tasks, we divided all probability values by 0.85. Inter-task communications follow the definitions of Table II in [14]. For each task we generated a WCET following the definitions of Tables IV and V in [14]. As stated in [14], in typical engine control applications there are between 30 and 60 cause-effect chains. In our experiments, on average, each task set has 38 cause-effect chains and 162 tasks. The number of periods per cause-effect chain is randomly chosen between the interval [1,3] following the definitions of Table VI in [14]. For each period that composes the cause-effect chain, there are 2 to 5 tasks with that same period (Table VII in [14]). Each cause-effect chain is composed of 2 to 15 tasks. Since MRT and MDA values are equivalent (Günzel et al. [9]), we shown in Figure 5a the results obtained for MRT. Note that in Figure 5a, we normalized the results with respect to the MRT obtained by the LET paradigm, i.e., the length of the communication intervals are equal to tasks period.



(a)



(b)

Fig. 5: Normalized MRT w.r.t LET

Figure 5a shows that applying our Algorithm 1 alone produces results equal or slightly worse than the ones obtained by Maia et al. [16], but without the complexity and need of iterating over the entire hyperperiod. With the current experimental setup, the combination of Algorithm 1 and precedence constraints was the one that obtained the best MRT values, an improvement of up to $\approx 17\%$ over Maia et al. [16]. For the results shown in Figure 5a, our method added an average of 13 precedence constraints per task set, being 4 the minimum and 72 the maximum. We noticed during the experiments that for some cause-effect chains, our methods obtains the same end-to-end latencies as Maia et al. [16], although the communication intervals have a different configuration.

Figure 5b shows that our methods benefit when tasks have fixed unique priorities that are not directly related to deadline or period. With the current experimental setup, Algorithm 1 reduces MRT values by $\approx 47\%$, on average, compared to the LET paradigm, while the combination of Algorithm 1 and precedence constraints reduced values, on average, by $\approx 48\%$.

7.2 Synthetically Generated Workloads

We randomly generated 1000 schedulable task sets, where we chose task periods and inter-task communication as in the previous experiment. However, this time we allowed tasks to have higher WCET, increased the number of possible periods per cause-effect chain from 3 to 5 and reduced the probability of single-rate cause-effect chains from 70% to 7%. The 63% difference was divided and added equally (63%/4=15.75%) to the other probability rates. The new probabilities for possible number of periods per chain are $\{1\colon 7\%, 2\colon 35.75\%, 3\colon 25.75\%, 4\colon 15.75\%, 5\colon 15.75\%\}$. We also increased the total number of tasks composing a cause-effect chain from 15 to 25. We chose randomly the amount of cause-effect chains per task set from interval $[10, 20]$. On average, each task set has 12 cause-effect chains and 111 tasks.

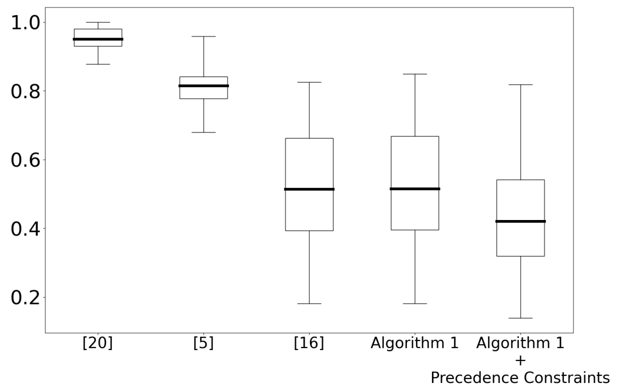


Fig. 6: Normalized MRT w.r.t LET

Figure 6 shows that both of our methods reduced the MRT values compared to the LET paradigm. For synthetic task sets, the improvements obtained by our methods over the LET paradigm were higher than the ones obtained for the automotive benchmark. Our methods further improved the MRT values up to $\approx 60\%$ and $\approx 70\%$ respectively. Similar to what we obtained before with the task sets from the benchmark, the combination of Algorithm 1 and precedence constraints improved the MRT values by up to $\approx 14\%$ compared to the values obtained by Maia et al. [16]. Without the complexity and need of iterating over the entire hyperperiod. Algorithm 1 alone obtained results equal or slightly worse than the ones obtained by Maia et al. [16], the difference for synthetic task sets was, on average, less than $\approx 1\%$. For the results shown in Figure 6, an average of 3 precedence constraints per task set, being 2 the minimum and 56 the maximum were added by our method.

8 Conclusion

In this paper, we proposed two methods to reduce the end-to-end latencies of multi-rate cause-effect chains by safely reconfiguring the communication intervals of the LET paradigm.
By exploiting harmonic properties of the task set, the first method shows how the correct phasing of communication interval boundaries can improve the end-to-end latencies of all cause-effect chains present in the system simultaneously. Compared to previous methods, it does not require an investigation of the complete hyperperiod resulting in a less complex analysis that provides better end-to-end latencies. Our second method optimizes the communication intervals by adding precedence constraints. This is achieved by a guided search targeted to reduce simultaneously the end-to-end latency of all chains present in the system. If needed, e.g., for legacy reasons, our methods does not have to be applied to the entire task set, but also to only a subset.
Experiments showed that for task sets based on the Real World Automotive Benchmarks presented by BOSCH [14] and synthetically generated, our methods result in lower end-to-end latencies when compared to the LET model and previous works. As future work, we plan to investigate the impact of our methods in cause-effect chains containing multiple execution models.

A Busy-Interval-Based Computation of Finishing Times

To prove results presented in Section 5, the finishing time of jobs under the influence of task phasing has to be calculated. To that end, the following lemmas, based on the busy-interval concept [15], can be utilized:

Lemma 3. *Given a job $\tau(i)$ of task τ. Let $b \leq r_{\tau(i)} \in \mathbb{R}$ be a time point such that higher priority tasks $\mathrm{hp}(\tau)$ are executed during $[b, r_{\tau(i)}]$ in schedule $\mathcal{S}_{max}$, and all jobs of $\mathrm{hp}(\tau)$ released before b finish at or before b in schedule $\mathcal{S}_{max}$. Then, $f^{\mathcal{S}_{max}}_{\tau(i)}$ is the earliest time $t \geq r_{\tau(i)}$ such that*

$$t \geq b + C_\tau + \sum_{\tau' \in \mathrm{hp}(\tau)} \eta_{\tau'}(b,t) \cdot C_{\tau'}, \tag{19}$$

where $\eta_{\tau'}(b,t) := \left\lceil \frac{\max(t-\phi_{\tau'},0)}{T_{\tau'}} \right\rceil - \left\lceil \frac{\max(b-\phi_{\tau'},0)}{T_{\tau'}} \right\rceil$ is the number of job releases of task τ' in the interval $[b,t)$.

Proof. By definition, $I_B := [b, f^{\mathcal{S}_{max}}_{\tau(i)}]$ is a level τ busy-interval in the sense that during I_B the system is busy executing jobs of $\{\tau\} \cup \mathrm{hp}(\tau)$. Furthermore, only jobs of $\{\tau\} \cup \mathrm{hp}(\tau)$ that are released during I_B are executed during I_B and all of them finish within I_B. Therefore, the end of I_B can be computed by finding the earliest $t \geq r_{\tau(i)}$ such that Equation (19) holds, where $\eta_{\tau'}(b,t)$ is the number of job releases of task τ' in the interval $[b,t)$.
To calculate $\eta_{\tau'}(b,t)$, we need to determine the number of job releases. Since, the first job of τ' is released at time $\phi_{\tau'}$, and then an additional job is released every

$T_{\tau'}$ time units, the total number of job releases within $(-\infty, t)$ can be calculated as $\left\lceil \frac{\max(t-\phi_{\tau'},0)}{T_{\tau'}} \right\rceil$. Consequently, the number of job releases in the interval $[b, t)$ is the number of job releases in $(-\infty, t)$ minus the number of job releases in $(-\infty, b)$, i.e., $\eta_{\tau'}(b,t) = \left\lceil \frac{\max(t-\phi_{\tau'},0)}{T_{\tau'}} \right\rceil - \left\lceil \frac{\max(b-\phi_{\tau'},0)}{T_{\tau'}} \right\rceil$. This proves the lemma.

While in general, the calculation of b can become quite complicated, the following lemma restricts to the examination of the finishing time $f_{\tau(1)}^{\mathcal{S}_{max}}$ of the first job $\tau(1)$ under a special setting identified in Section 5.

Lemma 4. *Let $\tau \in \Gamma$, such that $\phi_{\tau'} \in [0, \max_{\tau'' \in \mathrm{hp}(\tau')} f_{\tau''(1)}^{\mathcal{S}_{max}}]$ for all $\tau' \in \mathrm{hp}(\tau) \cup \{\tau\}$.*[6] *Then, $f_{\tau(1)}^{\mathcal{S}_{max}}$ is the earliest time $t \geq \phi_\tau$ such that:*

$$t \geq C_\tau + \sum_{\tau' \in \mathrm{hp}(\tau)} \left\lceil \frac{t - \phi_{\tau'}}{T_{\tau'}} \right\rceil \cdot C_{\tau'} \tag{20}$$

Proof. The condition $\phi_{\tau'} \in [0, \max_{\tau'' \in \mathrm{hp}(\tau')} f_{\tau''(1)}^{\mathcal{S}_{max}}]$ for all $\tau' \in \mathrm{hp}(\tau) \cup \{\tau\}$ ensures that the system is busy executing jobs of $\mathrm{hp}(\tau) \cup \{\tau\}$ during the interval $[0, \phi_\tau]$. Therefore, we can apply Lemma 3 with $b = 0$ and $i = 1$, i.e., $f_\tau^{\mathcal{S}_{max}}$ is the earliest time $t \geq r_{\tau(1)} = \phi_\tau$ such that $t = C_\tau + \sum_{\tau' \in \mathrm{hp}(\tau)} \eta_{\tau'}(0,t) \cdot C_{\tau'}$ with $\eta_{\tau'}(0,t) = \left\lceil \frac{\max(t-\phi_{\tau'},0)}{T_{\tau'}} \right\rceil - \left\lceil \frac{\max(0-\phi_{\tau'},0)}{T_{\tau'}} \right\rceil$. Since $\phi_{\tau'} \geq 0$, we know that $\left\lceil \frac{\max(0-\phi_{\tau'},0)}{T_{\tau'}} \right\rceil = 0$, i.e., $\eta_{\tau'}(0,t) = \left\lceil \frac{\max(t-\phi_{\tau'},0)}{T_{\tau'}} \right\rceil$. Furthermore,

$$\left\lceil \frac{\max(t - \phi_{\tau'}, 0)}{T_{\tau'}} \right\rceil = \left\lceil \frac{t - \phi_{\tau'}}{T_{\tau'}} \right\rceil \tag{21}$$

for $t \geq 0$ as show by case distinction:

- **Case 1**: If $t \geq \phi_{\tau'}$, then $\max(t - \phi_{\tau'}, 0) = t - \phi_{\tau'}$ and therefore (21) holds.
- **Case 2**: If $t \in [0, \phi_{\tau'})$, then $t - \phi_{\tau'} > -T_{\tau'}$, because otherwise $T_{\tau'} \leq \phi_{\tau'}$ which would mean that $D_{\tau'} \leq T_{\tau'} \leq \phi_{\tau'} \leq \max_{\tau'' \in \mathrm{hp}(\tau')} f_{\tau''(1)}^{\mathcal{S}_{max}}$. Specifically, the system would be busy executing jobs of tasks in $\mathrm{hp}(\tau')$ during an interval $[0, \max_{\tau'' \in \mathrm{hp}(\tau')} f_{\tau''(1)}^{\mathcal{S}_{max}}]$ of length $\geq D_{\tau'}$, i.e., τ' would not be schedulable. This contradicts the assumption in the system model that the task set is schedulable. That is, for Case 2, we conclude $t - \phi_{\tau'} > -T_{\tau'}$, and therefore Equation (21) holds.

Using Equation (21), we obtain $\eta_{\tau'}(0,t) = \left\lceil \frac{t-\phi_{\tau'}}{T_{\tau'}} \right\rceil$, which proves the lemma.

Acknowledgments

The work presented in this paper was supported in part by the Fonds National de la Recherche Luxembourg (FNR), the Deutsche Forschungs Gemeinschaft (DFG) through the Core-Inter project ReSAC, and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 865170) through the PropRT project.

[6] As in Section 5, we assume that $\max_{\tau'' \in \mathrm{hp}(\tau')} f_{\tau''(1)}^{\mathcal{S}_{max}} = 0$ if $\mathrm{hp}(\tau') = \emptyset$.

References

1. Audsley, N.C., Burns, A., Richardson, M.M., Tindell, K., Wellings, A.J.: Applying new scheduling theory to static priority pre-emptive scheduling. Softw. Eng. J. **8**(5), 284–292 (1993). https://doi.org/10.1049/SEJ.1993.0034, https://doi.org/10.1049/sej.1993.0034
2. Becker, M., Dasari, D., Mubeen, S., Behnam, M., Nolte, T.: Synthesizing job-level dependencies for automotive multi-rate effect chains. In: 2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA). IEEE (2016)
3. Biondi, A., Di Natale, M.: Achieving predictable multicore execution of automotive applications using the let paradigm. In: 2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). pp. 240–250. IEEE (2018)
4. Biondi, A., Pazzaglia, P., Balsini, A., Di Natale, M.: Logical execution time implementation and memory optimization issues in autosar applications for multicores. In: International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS) (2017)
5. Bradatsch, C., Kluge, F., Ungerer, T.: Data age diminution in the logical execution time model. In: International conference on Architecture of computing systems. pp. 173–184. Springer (2016)
6. Feiertag, N., Richter, K., Nordlander, J., Jonsson, J.: A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics. In: IEEE Real-Time Systems Symposium: 30/11/2009-03/12/2009. IEEE Communications Society (2009)
7. Günzel, M., Becker, M.: Optimal task phasing for end-to-end latency in harmonic and semi-harmonic automotive systems. In: 2025 IEEE 31th Real-Time and Embedded Technology and Applications Symposium (RTAS). IEEE (2025)
8. Günzel, M., Chen, K.H., Ueter, N., Brüggen, G.v.d., Dürr, M., Chen, J.J.: Compositional timing analysis of asynchronized distributed cause-effect chains. ACM Transactions on Embedded Computing Systems **22**(4), 1–34 (2023)
9. Günzel, M., Teper, H., Chen, K.H., von der Brüggen, G., Chen, J.J.: On the equivalence of maximum reaction time and maximum data age for cause-effect chains. In: 35th Euromicro Conference on Real-Time Systems (ECRTS 2023). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2023)
10. Hamann, A., Dasari, D., Kramer, S., Pressler, M., Wurst, F.: Communication centric design in complex automotive embedded systems. In: 29th Euromicro Conference on Real-Time Systems (ECRTS 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)
11. Henzinger, T.A., Horowitz, B., Kirsch, C.M.: Giotto: A time-triggered language for embedded programming. In: International Workshop on Embedded Software. pp. 166–184. Springer (2001)
12. Klaus, T., Becker, M., Schröder-Preikschat, W., Ulbrich, P.: Constrained data-age with job-level dependencies: How to reconcile tight bounds and overheads. In: 2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS). pp. 66–79. IEEE (2021)
13. Kordon, A., Tang, N.: Evaluation of the age latency of a real-time communicating system using the let paradigm. In: ECRTS 2020. vol. 165. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2020)
14. Kramer, S., Ziegenbein, D., Hamann, A.: Real world automotive benchmarks for free. In: 6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS). vol. 130 (2015)

15. Lehoczky, J.P.: Fixed priority scheduling of periodic task sets with arbitrary deadlines. In: RTSS. pp. 201–209. IEEE Computer Society (1990)
16. Maia, L., Fohler, G.: Reducing end-to-end latencies of multi-rate cause-effect chains in safety critical embedded systems. In: 12th European Congress on Embedded Real Time Software and Systems (ERTS 2024) (2024)
17. Maia, L., Fohler, G.: Decreasing utilization of systems with multi-rate cause-effect chains while reducing end-to-end latencies. In: 2025 28th International Symposium on Real-Time Distributed Computing (ISORC). IEEE (2025)
18. Mäki-Turja, J., Nolin, M.: Efficient implementation of tight response-times for tasks with offsets. Real-Time Systems **40**, 77–116 (2008)
19. Martí, P., Villa, R., Fuertes, J.M., Fohler, G.: On real-time control tasks schedulability. In: 2001 European control conference (ECC). pp. 2227–2232. IEEE (2001)
20. Martinez, J., Sañudo, I., Bertogna, M.: Analytical characterization of end-to-end communication delays with logical execution time. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **37**(11), 2244–2254 (2018)
21. Palencia, J.C., Harbour, M.G., Gutiérrez, J.J., Rivas, J.M.: Response-time analysis in hierarchically-scheduled time-partitioned distributed systems. IEEE Transactions on Parallel and Distributed Systems **28**(7), 2017–2030 (2016)
22. Palencia, J.C., Harbour, M.G.: Schedulability analysis for tasks with static and dynamic offsets. In: Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No. 98CB36279). pp. 26–37. IEEE (1998)
23. Pazzaglia, P., Biondi, A., Di Natale, M.: Optimizing the functional deployment on multicore platforms with logical execution time. In: 2019 IEEE Real-Time Systems Symposium (RTSS). pp. 207–219. IEEE (2019)
24. Syed, A., Fohler, G.: Efficient offline scheduling of task-sets with complex constraints on large distributed time-triggered systems. Real-Time Systems **55**, 209–247 (2019)
25. Tindell, K.: Adding time-offsets to schedulability analysis (1994)
26. Verucchi, M., Theile, M., Caccamo, M., Bertogna, M.: Latency-aware generation of single-rate dags from multi-rate task sets. In: 2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). pp. 226–238. IEEE (2020)
27. Wang, S., Li, D., Sifat, A.H., Huang, S.Y., Deng, X., Jung, C., Williams, R., Zeng, H.: Optimizing logical execution time model for both determinism and low latency. In: 2024 IEEE 30th Real-Time and Embedded Technology and Applications Symposium (RTAS). IEEE (2024)