Compositional Timing Analysis of Asynchronized Distributed Cause-effect Chains

MARIO GÜNZEL, TU Dortmund University
KUAN-HSUN CHEN, University of Twente
NIKLAS UETER, GEORG VON DER BRÜGGEN, MARCO DÜRR, and JIAN-JIA CHEN,
TU Dortmund University

Real-time systems require the formal guarantee of timing constraints, not only for the individual tasks but also for the end-to-end latency of data flows. The data flow among multiple tasks, e.g., from sensors to actuators, is described by a cause-effect chain, independent from the priority order of the tasks. In this article, we provide an end-to-end timing-analysis for cause-effect chains on asynchronized distributed systems with periodic task activations, considering the maximum reaction time (MRT) (i.e., the duration of data processing) and the maximum data age (MDA) (i.e., the worst-case data freshness). We first provide an analysis of the end-to-end latency on one local electronic control unit (ECU) that has to consider only the jobs in a bounded time interval. We extend our analysis to globally asynchronized systems by exploiting a compositional property to combine the local results. Throughout synthesized data based on an automotive benchmark as well as on randomized parameters, we show that our analytical results improve the state-of-the-art.

CCS Concepts: • **Computer systems organization → Embedded and cyber-physical systems**; **Real-time systems**;

Additional Key Words and Phrases: Cause-effect chains, maximum data age, maximum reaction time, compositional end-to-end analysis

ACM Reference format:
Mario Günzel, Kuan-Hsun Chen, Niklas Ueter, Georg von der Brüggen, Marco Dürr, and Jian-Jia Chen. 2023. Compositional Timing Analysis of Asynchronized Distributed Cause-effect Chains. *ACM Trans. Embedd. Comput. Syst.* 22, 4, Article 63 (July 2023), 34 pages.
https://doi.org/10.1145/3587036

1 INTRODUCTION

Industrial systems with real-time constrains require timeliness to ensure their correct functionality. Specifically, timing properties like end-to-end latencies are used to validate safety-critical

This result is part of a project (PropRT) that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 865170). This work has been supported by Deutsche Forschungsgemeinschaft (DFG), as part of Sus-Aware (Project no. 398602212).
Authors' addresses: M. Günzel, N. Ueter, G. von der Brüggen, M. Dürr, and J.-J. Chen, TU Dortmund University, Otto-Hahn-Str. 16, 44227 Dortmund, Germany; emails: mario.guenzel@tu-dortmund.de, niklas.ueter@tu-dortmund.de, georg.on-der-brueggen@tu-dortmund.de, marco.duerr@tu-dortmund.de, jian-jia.chen@tu-dortmund.de; K.-H. Chen, University of Twente, Zilverling (building no. 11), room 5037, Hallenweg 19, 7522NH Enschede, The Netherlands; email: k.h.chen@utwente.nl.

1539-9087/2023/07-ART63
https://doi.org/10.1145/3587036

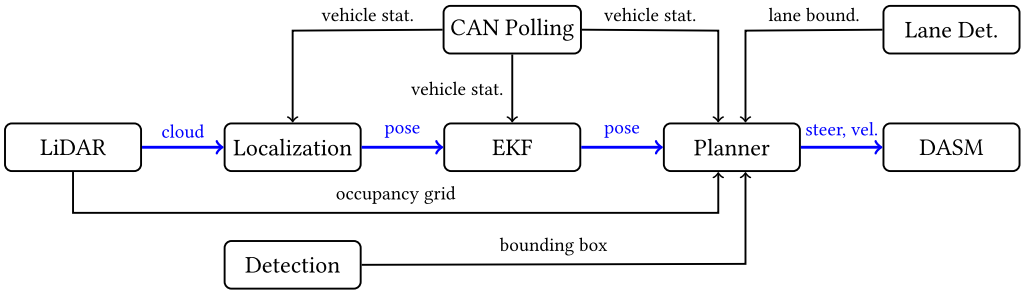


Fig. 1. An example application redrawn from the WATERS challenge 2019 [20]. An exemplary chain in the application is emphasized by the bold blue arrows.

tasks that have to perform a desired control within a certain time interval, e.g., when analyzing the interaction of **electronic control units** (**ECUs**) in a car.

A *cause-effect chain* is a strictly ordered set of tasks where the order describes data read and write dependencies. The WATERS industry challenge 2019 [20] provided an example, shown in Figure 1, that consists of the strictly ordered set of a lidar grabber task, a localization task, a sensor fusion task, a trajectory planner task, and a control task that also sets the actuators with velocity and steering signals. In this example, an instance (job) of the lidar grabber task reads information from the LiDAR sensor, builds a point cloud, and shares that point cloud data with a job of the localization task. Similarly, the computed data from the trajectory planner job is shared with a control and actuation job. In a so-called *end-to-end timing analysis*, the timing behavior of the function implemented by the task chain, i.e., the steering and velocity control based on the LiDAR sensory inputs, is analyzed with respect to ***maximum reaction time*** (***MRT***) *and* ***maximum data-age*** (**MDA**).

The *MRT* denotes the length of the longest time interval starting from the occurrence of an external cause (in the WATERS example new LiDAR sensor data) to the earliest time where this external cause is fully processed (actuated), i.e., the maximum *button to action delay*. The *MDA* denotes the length of the longest time interval starting with sampling a value to the last point in time where an actuation is based on this sampled value.

Most approaches in the literature that validate timing requirements of cause-effect chains can be classified into two categories: active approaches [9, 16, 28], which control the release of jobs in the subsequent tasks in the chain to ensure that the data is correctly written and read, and passive approaches [2, 3, 6, 10, 11, 13, 15, 17, 22, 27, 29], which analyze how the data is produced and consumed among the job of the recurrent tasks in the cause-effect chain. The approaches proposed in this work can be classified as passive approaches.

When all tasks in the system are mapped to one embedded device, they can be assumed synchronized as they have access to the same local clock. However, when tasks are mapped to multiple embedded devices, each device usually has its own clock with a different offset and jitter; a so-called globally asynchronized system. To enable a coherent collaboration among multiple distributed devices, clock synchronization techniques [23] enable one global system clock for all devices; a so-called globally synchronized system. Globally synchronized systems are usually easier to be optimized and analyzed, but the synchronization mechanism imposes strong dependencies (e.g., on the master node for clock synchronization), which makes the distributed system fragile against fault tolerance [6]. Therefore, accepting globally asynchronous clocks with offset and jitter enables distributed embedded systems that are *robust against imperfect synchronization of the architecture*. We call such systems **globally asynchronized locally synchronized** (**GALS**) systems.

Multiple results for the end-to-end timing analysis of distributed systems with different synchronization assumptions have been provided in the literature:

- —Davare et al. [10] provide an upper bound for the MRT of cause-effect chains for periodic task sets on GALS systems.
- —Dürr et al. [11] present two upper bounds, one for MRT and one for MDA, considering cause-effect chains for sporadic task sets on GALS systems.
- —Kloda et al. [22] provide an upper bound for the MRT of cause-effect chains for periodic task sets on globally synchronized systems.
- —Becker et al. [4] provide an upper bound for the MDA of cause-effect chains for periodic task sets on globally synchronized systems.

Since Dürr et al. [11] show that the MDA is less than or equal to the MRT, the upper bounds provided by Davare et al. [10] and Kloda et al. [22] also hold for the MDA. However, the analysis by Kloda et al. [22] is restricted to synchronous task releases, i.e., the first job of each task is released at time 0, and assumes that the **worst-case response time** (**WCRT**) of each task is known beforehand. Moreover, the article from Schlatow et al. [29] focuses on the analysis of MDA of harmonic task systems. Their analysis for non-harmonic cases is in fact more pessimistic than Davare's analysis [10], i.e., Equation (36) in [29] plus the WCRTs of the tasks in the chain is dominated by Davare's analysis [10]. Becker et al. [2, 3, 5] analyze end-to-end timing agnostic of the scheduler, i.e., only using information about the task release and deadline, with the goal to synthesize job-level dependencies to tighten timing guarantees. Their extension [4] analyzes the MDA with various levels of timing information. Recently, Gohari et al. [17] provided an analysis for the MDA under non-preemptive scheduling, whereas this work focuses on preemptive systems.

Dürr et al. [11] introduce *job chains* to describe data flow through a task set and to define MRT and MDA. In particular, they show that the MRT (MDA, respectively) of a schedule can be determined by computing the maximum length of immediate forward (backward, respectively) job chains. However, these definitions of MRT and MDA are limited to a sporadic task model where each task has a minimum and a maximum inter-arrival time. This work provides a definition that is not bounded to any specific task model.

Previous results [10, 11, 22] (implicitly) assume that MRT and data age are only measured when all tasks are already in the system, i.e., when all tasks have released their first job. Validation of this assumption is only possible by correctly accounting for the globally asynchronous clocks of the ECUs, which may be difficult to achieve or even impossible. Furthermore, such an assumption also prohibits the possibility of *compositional end-to-end timing analysis*, where a cause-effect chain is decomposed into smaller segments that can be analyzed separately with a *compositional property* to be used for analyzing the MRT and data age. We refine this assumption so that a compositional property can be achieved and there is no need for any assumption on global time.

It is worth noting that Dürr et al. [11] showed that their proposed end-to-end timing analyses for the sporadic task model analytically dominate the upper bound proposed by Davare et al. [10]. In this work, we leverage on the analytical upper bounds by Dürr et al. [11] as a backbone and improve them by showing that considering only a finite time window is sufficient for the analysis.

We provide an end-to-end analysis for single ECUs, in which predefined periodic tasks are scheduled under a fixed-priority preemptive scheduling policy. Subsequently, we utilize the compositional property to extend the analysis to the interconnected ECU scenario, in which multiple single ECUs are connected by an inter-communication infrastructure, e.g., **controller area network** (**CAN**) [8] or FlexRay [14]. For the interconnected scenario, we assume partitioned scheduling, i.e., there are individual periodic task sets for each ECU. Whereas tasks on a single ECU are scheduled using one synchronized clock, the clocks among different ECUs are usually not

synchronized. Such GALS are of high practical relevance. For instance, according to the FlexRay standard [14], the data communication cycle is divided into static segments (i.e., synchronous time-triggered communication) and dynamic segments (i.e., asynchronous event-driven communication). We note that, although the notion of *ECUs* is adopted from automotive systems, our work is not limited to automotive systems but can be extended to similar settings.

Contributions: We examine MRT and MDA of cause-effect chains for asynchronous periodic task sets on GALS distributed systems. Our main contributions are:

- In Section 4, we provide precise definitions of *MRT* and *MDA*. The underlying model only assumes recurrently released jobs with certain read- and write-operations. Hence, the definition is valid for all well-known task models (e.g., periodic tasks and sporadic tasks) and communication models (e.g., implicit communication and **logical execution time** (**LET**)). It covers the single ECU as well as the interconnected ECU scenario. In particular, in Section 4.3 we show that MRT and MDA allow a compositional property in form of the Cutting-Theorem.
- In Section 5.1, we provide a method to analyze MRT and MDA in the single ECU scenario for periodic tasks under preemptive fixed-priority scheduling. In particular, we show that a safe upper bound can be conducted from two extreme cases, i.e., applying only the **best-case execution time** (**BCET**) and the **worst-case execution time** (**WCET**) over a bounded time horizon.
- Section 5.2 extends the local analysis to interconnected ECUs and shows how to bound the time for communication between ECUs in a globally asynchronized distributed system.
- In Section 6, we discuss how the results of this work regarding the MDA can be applied to the maximum *reduced* data age, which is more common in the literature.
- We evaluate the proposed analysis for single and interconnected ECUs in Section 7, showing that it outperforms state-of-the-art analyses for both MRT and MDA. Moreover, we compare it with lower bounds for MDA and MRT and conclude that our bounds are close to the exact result.

This manuscript is based on a conference article [18] which focuses on a special scenario, assuming fixed execution time of a periodic task, i.e., the WCET is the same as the BCET. The solution presented in Section 5.1 is also an exact end-to-end analysis for the special scenario in the conference article [18] but can be applied to more general cases, in which the execution time of a task can be any value between its best-case and worst-case execution time.

2 SYSTEM MODEL

In this section, we introduce definitions and notation for the task model, the communication model, cause-effect chains, and job chains utilized in this work.

2.1 Jobs and Tasks

For a general definition of MRT and MDA, we rely on a very basic model of jobs and tasks that are executed on multiple ECUs. If the ECUs are not synchronized, i.e., their clocks are not aligned, then we take their clock shifts into account to compare the time of events on the ECUs on a global level. First, we introduce *jobs*, *schedules*, and *tasks*. We assume that there is no parallel execution of jobs on one ECU, i.e., each ECU is a uniprocessor system.

A *job* J is an instance of a program, which produces an output based on its input. It is released at time r_J and has to be executed for a certain amount of time $c_J \geq 0$ to finish. A *schedule* $\mathcal{S}$ specifies the execution behavior of jobs on the ECUs. If J is scheduled by $\mathcal{S}$, the start time (or start) of J is

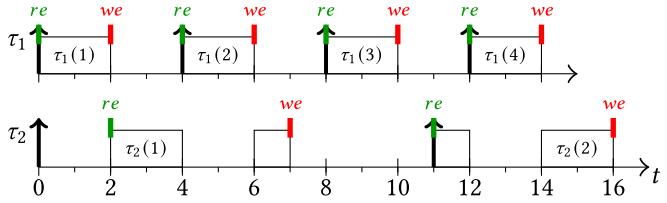


Fig. 2. Read- and write-events under implicit communication.

denoted $s_J^{\mathcal{S}}$ and the finishing time (or finish) of J is $f_J^{\mathcal{S}}$. For the sake of readability, we omit the index $\mathcal{S}$ for all definitions if the choice of a schedule is clear in the context. The aggregation of all jobs which are instances of the same program is called a *task*, denoted by τ. We assume that each task is assigned to one ECU, i.e., all jobs of one task are scheduled on the same ECU, and that the jobs aggregated to one task τ are countable. We denote the set of all tasks as $\mathbb{T}$ and the jobs of τ as $(\tau(m))_{m\in\mathbb{N}}$, with $0 \notin \mathbb{N}$. Furthermore, we assume that the task set $\mathbb{T}$ is finite.

Whereas the general definitions from Section 4 are valid for all kinds of task models, e.g., for periodic or sporadic tasks, the analysis in Section 5 is limited to *periodic tasks*. A periodic task is described by the tuple $\tau = (C_\tau^u, C_\tau^\ell, T_\tau, \phi_\tau) \in \mathbb{R}^4$, where C_τ^u and $C_\tau^\ell \geq 0$ are the WCET and the BCET of the task, respectively, $T_\tau > 0$ is the period, and $\phi_\tau \geq 0$ is the phase of the task. By definition, $C_\tau^u \geq C_\tau^\ell \geq 0$. The first job is released at time ϕ_τ. Afterward, τ recurrently releases a job every T_τ time units. More specifically, we have $r_{\tau(m)} = \phi_\tau + (m-1)\cdot T_\tau$ and $c_{\tau(m)} \in [C_\tau^u, C_\tau^\ell]$ for all $m \in \mathbb{N}$. The utilization of a task τ is defined by $U_\tau := \frac{C_\tau^u}{T_\tau}$. We assume that the total utilization $U_{\mathbb{T}} := \sum_{\tau\in\mathbb{T}} U_\tau$ of a task set $\mathbb{T}$ on a single ECU is at most 1. The maximal phase of $\mathbb{T}$ is denoted by $\Phi = \Phi(\mathbb{T}) := \max_{\tau\in\mathbb{T}} \phi_\tau$. The hyperperiod of $\mathbb{T}$ is $H = H(\mathbb{T}) := \operatorname{lcm}(\{T_\tau \mid \tau \in \mathbb{T}\})$, i.e., the least common multiple of all periods in the system. The existence of such hyperperiod is required for our analysis in Section 5.

During analysis, usually, only the task description is known instead of the actual jobs. Therefore, all schedules $\mathcal{S}$ that are compatible with the task specification must be considered. If the underlying scheduling algorithm is deterministic, then each schedule is uniquely determined by the release time and execution time of all jobs in the schedule. In this case, it is sufficient to consider all job sequences that are compatible with the task specification to cover all possible schedules. We call one of those job sequences for task τ a *job collection* for τ. Let $JC(\tau)$ be the set of all job collections of task τ. Since in each schedule $\mathcal{S}$ for each task one job collection is under consideration, we define the set of all job collections for $\mathbb{T}$ as the cartesian product of the sets of job collections for each task, i.e., $JC(\mathbb{T}) := \prod_{\tau\in\mathbb{T}} JC(\tau)$. One schedule $\mathcal{S}$ is then equivalent to one job collection $jc \in JC(\mathbb{T})$, and we write $\mathcal{S} = \mathcal{S}(jc)$. To distinguish jobs from different job collections, we denote by $\tau(m, jc)$ the mth job of a task τ in the job collection jc, and use $\tau(m)$ if the underlying job collection jc is clear in the context.

2.2 Communication Model

When jobs communicate, they receive (read) their input from a shared resource and hand over (write) their output to a shared resource. Jobs from the same tasks write to the same resource and messages are overwritten; that is, at any time only the latest output of a task is available. We denote the first *read-event* of a job J in the schedule $\mathcal{S}$ by $re_J^{\mathcal{S}}$, and we denote the last *write-event* of J in $\mathcal{S}$ by $we_J^{\mathcal{S}}$. For the sake of clear notation, we write re_J and we_J if the schedule $\mathcal{S}$ is obvious or irrelevant to determine the read- and write-events. We consider two common communication policies. One is called *implicit communication*, where the read- and write-events are aligned with the start and finish of the jobs, respectively (i.e., $re_J^{\mathcal{S}} = s_J^{\mathcal{S}}$ and $we_J^{\mathcal{S}} = f_J^{\mathcal{S}}$), as depicted in Figure 2.

The other one is based on the concept of *LET* [21]. To utilize LET, each task τ is equipped with a *relative deadline* D_τ. Each job J released by a task τ has an *absolute deadline* $d_J = r_J + D_\tau$. The read- and write-events of each job J are set to its release time and deadline, respectively (i.e., $re_J^{\mathcal{S}} = r_J$ and $we_J^{\mathcal{S}} = d_J$). Although LET is originally limited to a single ECU, Ernst et al. [12] provide a generalization to the interconnected ECU setup. If we utilize LET in the following, we consider only *feasible* schedules, in which each job finishes before its deadline, i.e., $f_J^{\mathcal{S}} \leq d_J$ for all jobs J in $\mathcal{S}$. Note that these two approaches provide a tradeoff: While implicit communication leads to shorter latencies, LET provides timing determinism [19].

We assume that the following (not very restrictive) requirements are met:

- The read- and write-events of the jobs of each task $\tau \in \mathbb{T}$ are ordered in the sense that $re^{\mathcal{S}}_{\tau(m)} < re^{\mathcal{S}}_{\tau(m+1)}$, $we^{\mathcal{S}}_{\tau(m)} < we^{\mathcal{S}}_{\tau(m+1)}$, and $re^{\mathcal{S}}_{\tau(m)} \leq we^{\mathcal{S}}_{\tau(m)}$ for all $m \in \mathbb{N}$.
- The sets $\{re^{\mathcal{S}}_{\tau(m)} | m \in \mathbb{N}\}$ and $\{we^{\mathcal{S}}_{\tau(m)} | m \in \mathbb{N}\}$ have no accumulation point, i.e., in each bounded time interval there are only finitely many read- and write-events.

We note that the above properties are fulfilled if we consider the most common task models; that is, periodic or sporadic tasks together with LET or implicit job communication.

We assume systems are composed of multiple ECUs. The communication infrastructure between different ECUs is modeled by additional *communication tasks* τ^c. Those are usual tasks in the sense of Section 2.1, where each job has the purpose of transferring data between ECUs. More specifically, jobs released by communication tasks read data from a shared resource of one ECU and write it to a shared resource of another ECU. We assume that the communication tasks are executed on dedicated components and do not impair the job execution of the non-communication tasks. For notational convenience, those dedicated components are treated as communication ECUs. This abstraction is valid for common inter-communication infrastructures like CAN [8] or FlexRay [14].

2.3 Cause-effect Chains

A *cause-effect chain* $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_k)$ describes the path of data through different programs by a finite sequence of tasks $\tau_i \in \mathbb{T}$. For example, if task τ_1 uses data provided by τ_3, then $E = (\tau_3 \rightarrow \tau_1)$. The task order in a cause-effect chain is not necessarily identical with the order given by the scheduling algorithm, i.e., a task may consume data produced by a lower-priority task. We denote by $|E|$ the number of tasks in E, where $|E| \geq 1$. Moreover, for $m \in \{1, \ldots, |E|\}$, $E(m)$ denotes the mth task of the cause-effect chain E. For example, let $E = (\tau_4 \rightarrow \tau_5 \rightarrow \tau_1)$, then $|E| = 3$, $E(1) = \tau_4$, $E(2) = \tau_5$, and $E(3) = \tau_1$. We note that cause-effect chains are inspired by event-chains of the AUTOSAR Timing Extensions [1], which represent chains of more general functional dependency.

To obtain data for the first task in a cause-effect chain, data may need to be *sampled*. We assume an implicit sampling rate, where the sampling for a cause-effect chain E happens at the read-event of each job of $E(1)$. Nevertheless, we can easily model any kind of sampling by adding *sampling tasks* to the system which read and write data at the time the sampling happens. Please note that the read- and write-events of the jobs of the sampling tasks need to fulfill the requirements assumed in the previous subsection. Such sampling tasks can usually be modeled as a task assigned to an additional ECU or as a task with WCET of 0, which means they do not affect the schedule $\mathcal{S}$.

We consider two types of cause-effect chains. *Local* cause-effect chains only contain tasks on a single ECU (with synchronized clock). The tasks of *interconnected* cause-effect chains are spread among multiple ECUs. These ECUs may either be *synchronized* or *asynchronized*, i.e., they have synchronized or asynchronized clocks. We note that the definitions in Section 4 are valid for all kinds of cause-effect chains; the distinction is only necessary for the analysis in Section 5.

2.4 Job Chains

The concept of job chains is essential to determine MRT and MDA. We adapt the definition from Dürr et al. [11] to our model with read- and write-events. Let E and $\mathcal{S}$ be a cause-effect chain and a schedule for $\mathbb{T}$, respectively.

Definition 1 (Job Chain). A job chain of E for $\mathcal{S}$ is a sequence $c^{E,\mathcal{S}} = (J_1, \ldots, J_{|E|})$ of data-dependent jobs of tasks in $\mathbb{T}$ with the following properties:

- J_i is a job of $E(i)$ for all $i \in \{1, \ldots, |E|\}$.
- Data is read by J_{i+1} after it is written by J_i in the schedule $\mathcal{S}$, i.e., $we_{J_i} \leq re_{J_{i+1}}$ for all $i \in \{1, \ldots, |E| - 1\}$.

Like Dürr et al. [11], we consider two types of job chains, namely, forward and backward job chains.

Definition 2 (Immediate Forward Job Chain). An *immediate forward job chain* is a job chain $c^{E,\mathcal{S}} = (J_1, \ldots, J_{|E|})$ where for all $i \in \{1, 2, \ldots, |E| - 1\}$ the read-event of the job J_{i+1} is the earliest after the write-event of the job J_i. That is, $J_{i+1} = \arg\min_{J \in E(i+1), re_J \geq we_{J_i}} re_J$.

Definition 3 (Immediate Backward Job Chain). An *immediate backward job chain* is a job chain $c^{E,\mathcal{S}} = (J_1, \ldots, J_{|E|})$ where for all $i \in \{|E|, |E| - 1, \ldots, 2\}$ the write-event of the job J_{i-1} is the last before the read-event of the job J_i. That is, $J_{i-1} = \arg\max_{J \in E(i-1), we_J \leq re_{J_i}} we_J$.

If we consider the schedule from Figure 2 with $E = (\tau_1 \rightarrow \tau_2)$, then $(\tau_1(1), \tau_2(1))$, $(\tau_1(2), \tau_2(2))$, and $(\tau_1(3), \tau_2(2))$ are immediate forward job chains, while $(\tau_1(1), \tau_2(1))$ and $(\tau_1(3), \tau_2(2))$ are immediate backward job chains.

3 PROBLEM DEFINITION

In this article, we analyze the *MRT* and the *MDA* of distributed cause-effect chains E in GALS systems. We assume that the task set associated with each ECU as well as communication tasks between the ECUs are given.

- **Input:** Some (interconnected) cause-effect chain E.
- **Output:** An upper bound on the MRT and an upper bound on the MDA of E.

To solve this problem, we (1) provide a local analysis in Section 5.1 under implicit communication policy, and (2) extend the analysis to several ECUs in Section 5.2.

4 MAXIMUM REACTION TIME AND MAXIMUM DATA AGE

This article presents an end-to-end timing analysis based on cause-effect chains, i.e., the time interval between the occurrence of a cause (external activity or sampling a sensor value) and a recognizable effect (finish processing the data or movement of an actuator) is determined. Such an end-to-end timing analysis guarantees the correct functionality of safety critical tasks within a given time frame. For control engineering, the *MRT* (How long does it take until an external cause is processed?) and the *MDA* (How old is the data used in an actuation?) of a cause-effect chain are of special interest.

4.1 Augmented Job Chains

Let E be a cause-effect chain and let S be a schedule for the task set $\mathbb{T}$. Data movement through the schedule $\mathcal{S}$ following the dependencies of E can be captured by a sequence of events from an external activity to actuation as shown in Figure 3: The (change of) data is the result of some

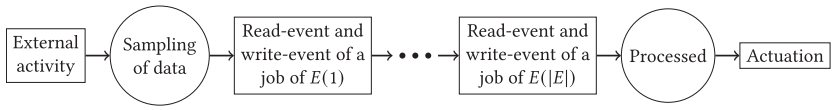


Fig. 3. Chain of events to trace one data stream of E.

external activity and is fed into the system by *sampling*. By our assumption in Section 2.3, the sampling coincides with the read-event of a job of the first task $E(1)$. When the first job in the sequence of events finishes execution, it writes the data to a shared resource. Afterward, the second job reads the data from that shared resource, processes it, and writes it again to a shared resource for the next task, and so on. When the last job in the sequence writes to a shared resource, the data is completely *processed* by the system. After that, an *actuation* can happen based on that data.

MDA and MRT are defined by Dürr et al. [11] using backward and forward job chains, respectively. In fact, job chains describe only the data stream from sampling until the data is processed. In this article, we cover the whole data stream by adding events for external activity and actuation. We call such extended job chains *augmented job chains*.

Definition 4 (Augmented Job Chain). An augmented job chain of E for schedule $\mathcal{S}$ is a sequence $c^{E,\mathcal{S}} = (z, J_1, \ldots, J_{|E|}, z')$, where $(J_1, \ldots, J_{|E|})$ is a job chain, and $z \leq re_{J_1}$ and $z' \geq we_{J_{|E|}}$ are time instants of an external activity and an actuation, respectively.

We denote by $c^{E,\mathcal{S}}(k)$ the kth entry of the augmented job chain $c^{E,\mathcal{S}}$. To be precise, $c^{E,\mathcal{S}}(1) = z$, $c^{E,\mathcal{S}}(|E| + 2) = z'$, and $c^{E,\mathcal{S}}(k) = J_{k-1}$ for $2 \leq k \leq |E| + 1$. To describe the time from external activity to actuation for one data stream, we define the *length* $\ell(c^{E,\mathcal{S}})$ of an augmented job chain $c^{E,\mathcal{S}}$ as

$$\ell(c^{E,\mathcal{S}}) := c^{E,\mathcal{S}}(|E| + 2) - c^{E,\mathcal{S}}(1) = z' - z. \tag{1}$$

In the following we omit the indices S and E of job chains if they are clear in the context.

The MRT bounds the time from external activity to the instant where data is completely processed by the system. We omit the time between the processed-event and actuation, by only considering augmented job chains where the actuation is the time of the processed-event, i.e., $z' = we_{J_{|E|}}$. The longest time from external activity to sampling occurs if the external activity takes place directly after the previous sampling event. Hence, we construct *immediate forward augmented job chains*, to determine the MRT, in the following way:

Definition 5 (Immediate Forward Augmented Job Chain). An *immediate forward augmented job chain* $\vec{c}_m^{E,\mathcal{S}}$ is the unique augmented job chain $(z, J_1, \ldots, J_{|E|}, z')$, such that:

- The external activity happens directly after the mth sampling, i.e., $z = re_{E(1)(m)}$.
- The sampling happens at the next read-event of $E(1)$, i.e., $J_1 = E(1)(m + 1)$.
- The sequence $(J_1, \ldots, J_E)$ is an immediate forward job chain for E in $\mathcal{S}$.
- The actuation is set to the time where the data is processed, i.e., $z' = we_{J_{|E|}}$.

For each $m \in \mathbb{N}$ there is an immediate forward augmented job chain. Comparing them, as done in the next subsection, yields the definition of MRT.

On the other hand, the MDA bounds the time from the sampling of data to an actuation based on that sampling. In the worst case, the actuation based on the data processed at a certain time happens directly before the next processed-event.

Definition 6 (Immediate Backward Augmented Job Chain). An *immediate backward augmented job chain* $\overleftarrow{c}_m^{E,\mathcal{S}}$ is the unique augmented job chain $(z, J_1, \ldots, J_{|E|}, z')$, such that:

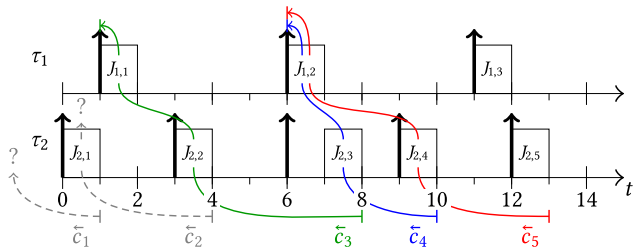


Fig. 4. An example of backward augmented job chains under implicit communication. The cause-effect chain under analysis is $E = (\tau_1 \rightarrow \tau_2)$.

- —The actuation happens directly before the mth processed-event, i.e., $z' = we_{E(|E|)(m)}$.
- —The processed-event happens at the previous write-event of $E(|E|)$, i.e., $J_{|E|} = E(|E|)(m-1)$.
- —The sequence $(J_1, \ldots, J_E)$ is an immediate backward job chain for E in $\mathcal{S}$.
- —The external activity is set to the time where the data is sampled, i.e., $z = re_{J_1}$.

Please note that there is not necessarily an immediate backward augmented job chain for all $m \in \mathbb{N}$. We call such chains *incomplete backward augmented job chains*. As no data is read for an incomplete backward augmented job chain, it is ignored when analyzing the data age. For brevity, the length of incomplete augmented job chains is set to 0.

Example 7 (Backward Augmented Job Chain Determination). Figure 4 shows a single ECU schedule with two periodic tasks $\tau_1 = (C^u_{\tau_1} = 1, C^\ell_{\tau_1} = 1, T_{\tau_1} = 5, \phi_{\tau_1} = 1)$ and $\tau_2 = (C^u_{\tau_2} = 1, C^\ell_{\tau_2} = 1, T_{\tau_2} = 3, \phi_{\tau_2} = 0)$. We assume implicit job communication, i.e., data is read at the start of each job and written at the finishing time of each job. The determination of the direct backward augmented job chain for $\overleftarrow{c}_5$ of cause-effect chain $E = (\tau_1 \rightarrow \tau_2)$ starts with 5-th write-event of the last task in the cause-effect chain at time 13. The backward job chain included in $\overleftarrow{c}_5$ is $(J_{1,2}, J_{2,4})$, and sampling is set to 6, which is the read-event of $J_{1,2}$. This leads to $\overleftarrow{c}_5 = (6, J_{1,2}, J_{2,4}, 13)$. Similar to the described procedure, $\overleftarrow{c}_4$ and $\overleftarrow{c}_3$ are determined. A special case occurs if we consider $\overleftarrow{c}_1$ or $\overleftarrow{c}_2$. The immediate backward augmented job chain $\overleftarrow{c}_1$ is incomplete, since there is no write-event of a job of τ_2 before $we_{J_{2,1}} = 1$. Moreover, $\overleftarrow{c}_2$ is incomplete as well since there is no immediate backward job chain with the second entry $J_{2,1}$.

We note that immediate forward augmented job chains and immediate backward augmented job chains are already uniquely determined by their corresponding job chain. The auxiliary entries for external activity and actuation are included for the simplicity of calculations.

4.2 Definition of MRT and MDA

Let E be a cause-effect chain and let $\mathcal{S}$ be a schedule with task set $\mathbb{T}$. With respect to Figure 3,

- —reaction time is the time from an external activity until the data is processed, and
- —data age is the time from a sampling of data to actuation based on that data.

We note that our definition differs from other definitions in the literature. For example, the data age definition by Dürr et al. [11] (denoted as the maximum *reduced* data age in this article) only covers the time until the data is processed. Our definition of MDA is necessary to obtain a compositional property in Section 4.3; that is, our definition ensures that the MDA of a cause-effect chain E is the sum of the MDA of the segments of E. However, in Section 6, we explain how the compositional property and local analysis can be applied to the reduced data age as well.

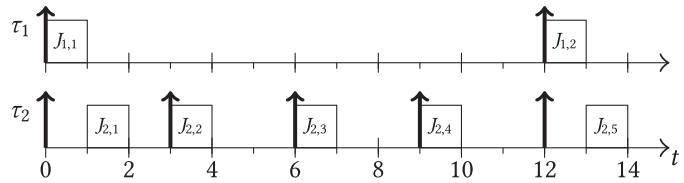


Fig. 5. Under implicit communication, the second task has slightly shifted read-event.

Similar to [11], we could define MRT (data age) as the supremum of the length of *all* immediate forward (backward) augmented job chains. However, due to shifting of the first read-event, e.g., induced by phases, the reaction time might become arbitrarily large:

Example 8 We consider a task set $\mathbb{T} = \{\tau_1, \tau_2\}$ with cause-effect chain $E = (\tau_1 \to \tau_2)$ and $re_{\tau_1(1)} = 0, re_{\tau_2(1)} = x$. The immediate forward augmented job chain $\vec{c}_1^{E,\mathcal{S}}$ has a length of at least x.

A solution to avoid this counterintuitive behavior is to only consider immediate forward (backward) augmented job chains $c^{E,\mathcal{S}}$ which start when all relevant tasks are in the system, i.e.,

$$c^{E,\mathcal{S}}(1) \geq \mathrm{Re}(E,\mathcal{S}) := \max_{i=1,\dots,|E|} re^{\mathcal{S}}_{E(i)(1)}. \tag{2}$$

This is similar to the (implicit) assumption by Dürr et al. [11] to measure MRT and data age only when all tasks are already in the system. However, due to this assumption, a slight shift of only one read-event might exclude multiple augmented job chains from consideration:

Example 9. For the schedule from Figure 5, if the jobs adhere implicit communication, the read-event of the first job of τ_2 is slightly shifted. With the approach from Equation (2), for a cause-effect chain $E(\tau_1 \to \tau_2)$ only augmented job chains c with $c(1) \geq 12$ would be considered, although $\vec{c}_1, \overleftarrow{c}_2, \overleftarrow{c}_3, \overleftarrow{c}_4$, and $\overleftarrow{c}_5$ should be included.

Hence, we only consider augmented job chains $c^{E,\mathcal{S}}$, if all tasks have their first read-event until the next read-event of $E(1)$ after $c^{E,\mathcal{S}}(1)$. We call these augmented job chains *valid*.

Definition 10 (Valid). Let $c^{E,\mathcal{S}} = (z, J_1, \dots, J_{|E|}, z')$ be some immediate forward or immediate backward augmented job chain for the cause-effect chain E in the schedule $\mathcal{S}$. Let $p \in \mathbb{N}$, such that $z = re_{E(1)(p)}$ holds. We call $c^{E,\mathcal{S}}$ *valid* if and only if $re_{E(1)(p+1)} > \mathrm{Re}(E,\mathcal{S})$.

We are now prepared to define the MRT and MDA.

Definition 11 (MRT and Data Age). For a cause-effect chain E with schedule $\mathcal{S}$ we define the schedule-specific *MRT* and *MDA* by

$$\mathrm{MRT}(E,\mathcal{S}) := \sup\left\{\ell\left(\vec{c}_m^{E,\mathcal{S}}\right) \,\middle|\, m \in \mathbb{N}, \vec{c}_m^{E,\mathcal{S}} \text{ valid}\right\}, \tag{3}$$

$$\mathrm{MDA}(E,\mathcal{S}) := \sup\left\{\ell\left(\overleftarrow{c}_m^{E,\mathcal{S}}\right) \,\middle|\, m \in \mathbb{N}, \overleftarrow{c}_m^{E,\mathcal{S}} \text{ valid}\right\}, \tag{4}$$

where the length ℓ of an event-chain is defined as in Equation (1).

Please note that this definition holds for all types of task sets and communication policies since the critical part is offloaded to the determination of the read- and write-events.

We also formulate a definition for MRT and data age which is not bounded to a specific schedule. If the procedure to pull job releases and execution times from a task set is specified, e.g., the task sets are periodic or sporadic, and if the scheduling algorithm is known beforehand, then this characterizes all possible schedules. Additionally, if the read- and write-events are uniquely

determined by the schedule, e.g., by following implicit communication or LET, then we define the *overall MRT* and *MDA*

$$\mathrm{MRT}(E) := \sup_{\mathcal{S}} \mathrm{MRT}(E, \mathcal{S}), \tag{5}$$

$$\mathrm{MDA}(E) := \sup_{\mathcal{S}} \mathrm{MDA}(E, \mathcal{S}), \tag{6}$$

by the supremum over all possible schedules $\mathcal{S}$. Furthermore, if the scheduling algorithm is deterministic, then there is a one-to-one mapping between the set of job collections $JC(\mathbb{T})$ and the set of schedules. In this case, the MRT and MDA can be formulated as

$$\mathrm{MRT}(E) = \sup_{jc \in JC(\mathbb{T})} \mathrm{MRT}(E, \mathcal{S}(jc)), \tag{7}$$

$$\mathrm{MDA}(E) = \sup_{jc \in JC(\mathbb{T})} \mathrm{MDA}(E, \mathcal{S}(jc)). \tag{8}$$

In their Theorem 6.2, Dürr et al. [11] prove that the data age is bounded by the reaction time for their system model. We note that even for this generalized definition

$$\mathrm{MDA}(E, \mathcal{S}) \leq \mathrm{MRT}(E, \mathcal{S}) \tag{9}$$

holds for all possible schedules $\mathcal{S}$: Let $\overleftarrow{c}_m^{E,\mathcal{S}}$ with $m \in \mathbb{N}$ be some valid immediate backward augmented job chain. Furthermore, let $p \in \mathbb{N}$ such that $\overleftarrow{c}_m^{E,\mathcal{S}}(1)$ coincides with the read-event of the pth job of task $E(1)$, that is, $\overleftarrow{c}_m^{E,\mathcal{S}}(1) = re_{E(1)(p)}$. Similar to the proof of Theorem 6.2 in [11] we show that

$$\ell\left(\overleftarrow{c}_m^{E,\mathcal{S}}\right) \leq \ell\left(\overleftarrow{c}_p^{E,\mathcal{S}}\right), \tag{10}$$

which is clearly upper bounded by the reaction time. Applying the supremum over all valid immediate backward job chains concludes the result. We note that also $\mathrm{MDA}(E) \leq \mathrm{MRT}(E)$ holds since Equation (9) holds for all possible schedules $\mathcal{S}$.

4.3 Cutting of Augmented Job Chains

One essential ingredient to apply a local analysis to the interconnected case in Section 5.2 is to cut the cause-effect chain into smaller (local) parts. Our definition of MRT and MDA enables the possibility to deduce upper bounds by determining MRT and MDA on the smaller segments, respectively. In this section, we prove that this compositional property holds for any task or communication model.

Theorem 12 (Cutting). *Let $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_{|E|})$ be any cause-effect chain. Furthermore, let $k \in \{1, \ldots, |E| - 1\}$ be some integer. For the cause-effect chains $E_1 := (\tau_1 \rightarrow \cdots \rightarrow \tau_k)$ and $E_2 := (\tau_{k+1} \rightarrow \cdots \rightarrow \tau_{|E|})$ holds that*

$$\mathrm{MRT}(E, \mathcal{S}) \leq \mathrm{MRT}(E_1, \mathcal{S}) + \mathrm{MRT}(E_2, \mathcal{S}), \tag{11}$$

$$\mathrm{MDA}(E, \mathcal{S}) \leq \mathrm{MDA}(E_1, \mathcal{S}) + \mathrm{MDA}(E_2, \mathcal{S}), \tag{12}$$

for any schedule $\mathcal{S}$.

The proof of the Cutting-Theorem relies on cutting immediate forward (backward) augmented job chains into smaller immediate forward (backward) augmented job chains, such that their combined length is at least the length of the initial immediate forward (backward) augmented job chain. Figure 6 shows the concept for immediate *backward* augmented job chains, assuming that

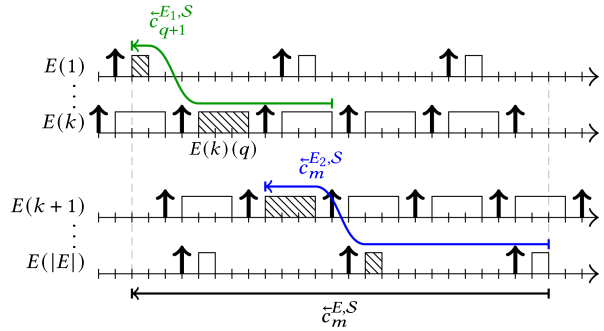


Fig. 6. Cutting one immediate backward augmented job chain $\overleftarrow{c}_m^{E,\mathcal{S}}$ into two as in the proof of Theorem 12 (Cutting).

jobs adhere implicit communication. We see that $\ell(\overleftarrow{c}_m^{E,\mathcal{S}}) \leq \ell(\overleftarrow{c}_{q+1}^{E_1,\mathcal{S}}) + \ell(\overleftarrow{c}_m^{E_2,\mathcal{S}})$. The jobs in the sequence of $\overleftarrow{c}_m^{E,\mathcal{S}}$, marked with the pattern, are distributed among $\overleftarrow{c}_{q+1}^{E_1,\mathcal{S}}$ and $\overleftarrow{c}_m^{E_2,\mathcal{S}}$. Only the events for external activity and actuation have to be determined properly.

Proof of Theorem 12 (Cutting). We first prove Equation (12). By definition, $\mathrm{MDA}(E,\mathcal{S})$ is the supremum of the length of all valid immediate backward augmented job chains. We consider some valid immediate backward augmented job chain $\overleftarrow{c}_m^{E,\mathcal{S}} = (re_{J_1}, J_1, \ldots, J_{|E|}, z')$ with $m \in \mathbb{N}$. Let $q \in \mathbb{N}$, such that J_k is the qth write-event of task $E(k)$, i.e., $E(k)(q) = J_k$. This scenario is depicted in Figure 6. By the definition of immediate backward augmented job chains, the write-event of $E(k)(q+1)$ occurs after the read-event of J_{k+1}, i.e., $\tilde{z}' := we_{E(k)(q+1)} > re_{J_{k+1}}$. Furthermore, $(re_{J_1}, J_1, \ldots, J_k, \tilde{z}') = \overleftarrow{c}_{q+1}^{E_1,\mathcal{S}}$ and $(re_{J_{k+1}}, J_{k+1}, \ldots, J_{|E|}, z') = \overleftarrow{c}_m^{E_2,\mathcal{S}}$ are immediate backward augmented job chains. They are both valid since $re_{J_{k+1}} \geq re_{J_1}$ and since $\overleftarrow{c}_m^{E,\mathcal{S}}$ is valid. We obtain

$$\ell\left(\overleftarrow{c}_m^{E,\mathcal{S}}\right) = z' - re_{J_1} \leq z' - re_{J_{k+1}} + \tilde{z}' - re_{J_1} = \ell\left(\overleftarrow{c}_{q+1}^{E_1,\mathcal{S}}\right) + \ell\left(\overleftarrow{c}_m^{E_2,\mathcal{S}}\right) \leq \mathrm{MDA}(E_1,\mathcal{S}) + \mathrm{MDA}(E_2,\mathcal{S}).$$

Applying the supremum yields the result from Equation (12).

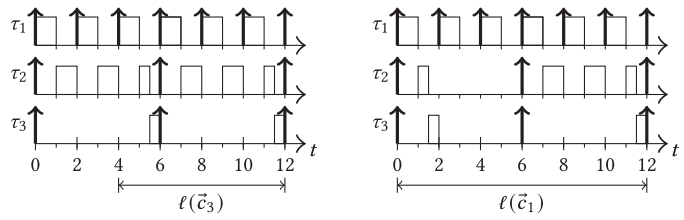
Analogously, we prove Equation (11). By definition, $\mathrm{MRT}(E,\mathcal{S})$ is the supremum of the length of all valid immediate forward augmented job chains. Let $\vec{c}_m^{E,\mathcal{S}} = (z, J_1, \ldots, J_{|E|}, we_{J_{|E|}})$ with $m \in \mathbb{N}$ be some valid immediate forward augmented job chain. Furthermore, let $p \in \mathbb{N}$ such that J_{k+1} is the p-th job of $E(k+1)$, i.e., $J_{k+1} = E(k+1)(p)$. By definition of immediate forward augmented job chains, the read-event of $E(k+1)(p-1)$ occurs before the write-event of J_k, i.e., $\tilde{z} := re_{E(k+1)(p-1)} < we_{J_k}$.

Since it is not clear directly, we shortly discuss the existence of the job $E(k+1)(p-1)$. We know that $E(k+1)(p)$ exists, i.e., $p \in \mathbb{N}$. It remains to show that $p \neq 1$. We know that, by definition of an immediate forward augmented job chain, $re_{E(k+1)(p)} \geq re_{J_1} > \mathrm{Re}(E,\mathcal{S})$ since $\vec{c}_m^{E,\mathcal{S}}$ is valid. If p would be 1, then $\mathrm{Re}(E,\mathcal{S}) \geq re_{E(k+1)(p)}$ which contradicts $re_{E(k+1)(p)} > \mathrm{Re}(E,\mathcal{S})$. This proves the existence of $E(k+1)(p-1)$.

With the definition of $\tilde{z}$ from above, we define two immediate forward augmented job chains $(z, J_1, \ldots, J_k, we_{J_k}) = \vec{c}_m^{E_1,\mathcal{S}}$ and $(\tilde{z}, J_{k+1}, \ldots, J_{|E|}, we_{J_{|E|}}) = \vec{c}_{p-1}^{E_2,\mathcal{S}}$. The augmented job chain $\vec{c}_m^{E_1,\mathcal{S}}$ is valid since the start coincides with the one of $\vec{c}_m^{E,\mathcal{S}}$ and since $\mathrm{Re}(E,\mathcal{S}) \geq \mathrm{Re}(E_1,\mathcal{S})$. The augmented job chain $\vec{c}_{p-1}^{E_2,\mathcal{S}}$ is also valid since $re_{E(k+1)(p)} \geq re_{J_1} > \mathrm{Re}(E,\mathcal{S}) \geq \mathrm{Re}(E_2,\mathcal{S})$. Hence,

$$\ell\left(\vec{c}_m^{E,\mathcal{S}}\right) = we_{J_{|E|}} - z \leq we_{J_{|E|}} - \tilde{z} + we_{J_k} - z = \ell\left(\vec{c}_m^{E_1,\mathcal{S}}\right) + \ell\left(\vec{c}_{p-1}^{E_2,\mathcal{S}}\right) \leq \mathrm{MRT}(E_1,\mathcal{S}) + \mathrm{MRT}(E_2,\mathcal{S}).$$

Applying the supremum yields the result from Equation (11). □



(a) Every task executes its WCET. (b) Early completion of task τ_2.

Fig. 7. Two schedules of jobs released periodically by 3 tasks. For $E = (\tau_1 \rightarrow \tau_3)$, early completion of the first job of τ_2 leads to a larger immediate forward augmented job chain.

Since Equations (11) and (12) hold for all schedules $\mathcal{S}$, the Cutting-Theorem does also hold for the overall MRT and overall MDA, i.e., $\text{MRT}(E) \leq \text{MRT}(E_1) + \text{MRT}(E_2)$ and $\text{MDA}(E) \leq \text{MDA}(E_1) + \text{MDA}(E_2)$. This compositional property can deal with clock-shifts by cutting at those positions where clock shifts occur.

5 ANALYSIS OF END-TO-END LATENCIES OF PERIODIC TASKS

In this section, we assume that the tasks on each ECU are scheduled according to preemptive fixed-priority scheduling. That is, on each ECU the tasks have a static priority-ordering and at each time the pending job of the task with the highest priority is executed. Our objective is to determine the MRT and the MDA of such systems.

Consider a schedule $\mathcal{S}$ of a periodic task system $\mathbb{T}$ and a cause-effect chain E of $\mathbb{T}$. If a recurrent pattern of the read- and write-events in $\mathcal{S}$ can be observed, then it suffices to analyze a limited time window to compute MRT and MDA. The way to achieve a recurrent pattern of read- and write-events depends on the communication policy.

For **LET**, the release pattern of all jobs repeats each hyperperiod after the maximal phase $\Phi := \max_\tau \phi_\tau$. Therefore, the read- and write-events repeat each hyperperiod after the maximal first read, which is at the maximal phase Φ, as well. Furthermore, all immediate forward and immediate backward augmented job chains with external activity at or after Φ are valid. In this case, it suffices to simulate all immediate backward and immediate forward augmented job chains with event for external activity during $[0, \Phi + H)$ and compute the maximum value among the length of all those valid augmented job chains. Kordon and Tang [24] compute the MDA on single ECU systems efficiently for LET, using this procedure as a backbone.

For **implicit communication**, the read- and write-events depend significantly on the execution time of the jobs under analysis. Furthermore, the read- and write-events might change when additional tasks are released. Due to this behavior, the pattern of read- and write-events in $\mathcal{S}$ does not repeat after $\Phi + H$. However, in Section 5.1.1 we show that the pattern of minimal and maximal read- and write-events for each job repeats after $\Phi + 2H$, and that it can be determined by simulating the schedule with the worst-case and with the best-case execution time in a bounded interval. This information can then be exploited to obtain an upper bound on the MDA and MRT. Please note that it is not sufficient to simulate the MDA and the MRT for the job collection where each job executes the WCET, as depicted in Figure 7. The remaining part of this section considers implicit communication.

5.1 Local Analysis

We assume that the cause-effect chain E under analysis is local, i.e., it contains only tasks on one (synchronized) ECU. Moreover, the tasks adhere to the implicit communication policy and

all schedules are obtained by a fixed-priority preemptive scheduling algorithm with fixed task priorities. For the sake of simplicity, we consider $\mathbb{T}$ to contain only tasks from one ECU as well, as the tasks from other ECUs have no impact on the scheduling behavior of the tasks on that ECU.

In the conference version [18] of this manuscript, the local analysis is achieved by enumerating possible immediate backward and forward augmented job chains, assuming a fixed execution time for each periodic task, i.e., the WCET is the same as the BCET. Our new analysis is applicable for more general cases, in which the execution time of a task can be any value between its best-case and worst-case execution time. It requires three steps:

(1) We calculate upper and lower bounds for both the read- and write-events of each job by simulating the schedule two times, once with the tasks WCETs and once with the BCETs.
(2) Based on these upper and lower bounds, we bound the length of all immediate forward and immediate backward augmented job chains.
(3) We bound the MDA and the MRT.

In addition to the generality provided in the analysis, we also show in Corollary 29 that our new analysis remains exact for the special scenario discussed in the conference version [18].

5.1.1 Step 1: Obtain Bounds for Read- and Write-events. Since under implicit communication the read- and write-events coincide with start and finish of the jobs, it is sufficient to examine those. Intuitively if the execution time of any job is increased, the interference on the other jobs does not decrease, and therefore the start and finish of all jobs cannot be decreased as well. Hence, the latest (earliest, respectively) start and finish of a job are achieved if all jobs execute their worst-case (best-case, respectively) execution time. Formally, we state the following two Propositions and refer for a rigorous proof to Appendix A.

Proposition 13. *Let jc_{max} be the job collection of a set $\mathbb{T}$ of periodic tasks where all jobs execute according to their WCET. Consider the mth job $J = \tau(m, jc_{max})$ of a task $\tau \in \mathbb{T}$ in the job collection jc_{max}. Then, for the preemptive fixed-priority schedule $\mathcal{S}(jc_{max})$, the starting time and finishing time of J in $\mathcal{S}(jc_{max})$ are **upper bounds** on the starting time and finishing time of the mth job of τ for* any *job collection, respectively. That is, for any other job collection $jc \in JC(\mathbb{T})$, we have $s_J^{\mathcal{S}(jc_{max})} \geq s_{\tau(m,jc)}^{\mathcal{S}(jc)}$ and $f_J^{\mathcal{S}(jc_{max})} \geq f_{\tau(m,jc)}^{\mathcal{S}(jc)}$.*

Proposition 14. *Let jc_{min} be the job collection of a set $\mathbb{T}$ of periodic tasks where all jobs execute according to their BCET. Consider the mth job $J = \tau(m, jc_{min})$ of a task $\tau \in \mathbb{T}$ in the job collection jc_{min}. Then, for the preemptive fixed-priority schedule $\mathcal{S}(jc_{min})$, the starting time and finishing time of J in $\mathcal{S}(jc_{min})$ are **lower bounds** on the starting time and finishing time of the mth job of τ for* any *job collection, respectively. That is, for any other job collection $jc \in JC(\mathbb{T})$, we have $s_J^{\mathcal{S}(jc_{min})} \leq s_{\tau(m,jc)}^{\mathcal{S}(jc)}$ and $f_J^{\mathcal{S}(jc_{min})} \leq f_{\tau(m,jc)}^{\mathcal{S}(jc)}$.*

Under implicit communication, the starting time and the finishing time coincide with the read- and write-events. Hence, the lower and upper bounds for starting time and finishing time from Propositions 13 and 14 are lower and upper bounds for the read- and write-events as well. We conclude that for each job we can provide upper and lower bounds for the read- and write-events by simulating the schedule two times until the job finishes: Once when all jobs execute their WCET and once when all jobs execute their BCET.

Since for each task, there are infinitely many jobs, we cannot simulate the schedule for each job individually. Therefore, in the following, we show that it is sufficient to simulate the schedule for a finite time window since the release pattern repeats. The following considerations are based on the work of Leung and Whitehead [26]. Their proofs cannot be used directly, since they create a

new schedule (they call it a partial schedule) and show that this one repeats. We need to show that even the original schedule repeats.

In the following, let $\mathcal{S}'$ be the fixed-priority schedule of the task set $\mathbb{T} = \{\tau_1, \ldots, \tau_n\}$ on one ECU where either the execution of all jobs is fixed to their worst-case, i.e., $\mathcal{S}' = \mathcal{S}(jc_{max})$, or the execution of all jobs is fixed to their best-case, i.e., $\mathcal{S}' = \mathcal{S}(jc_{min})$. We denote by $C_1, \ldots, C_n$ the fixed execution times of all jobs of the tasks $\tau_1, \ldots, \tau_n$. Without loss of generality, we assume that the tasks' indices are assigned according to their priority, i.e., τ_i has a higher priority than τ_j if and only if $i < j$. For a time instant t, we denote by $exec(\mathcal{S}', t)$ the tuple $(s_{1,t}, \ldots, s_{n,t})$ where each $s_{i,t}$ is the amount of time that jobs of τ_i have been executed since their last release. Similar to the proof of Leung and Whitehead [26, Lemma 3.3], we show the following.

Lemma 15. *For all $t \geq \Phi$ the relation $exec(\mathcal{S}', t) \geq exec(\mathcal{S}', t + H)$ (component-wise) holds.*

Proof. For a proof by contradiction, we assume that there are some $t \geq \Phi$ and $i \in \{1, \ldots, n\}$ such that $s_{i,t} < s_{i,t+H}$. We show that in this case infinitely many tasks τ_j have a time instant

$$t_j \geq \phi_{\tau_j} \text{ with } s_{j,t_j} < s_{j,t_j+H}. \tag{13}$$

This contradicts the fact that $\mathbb{T}$ is finite.

By assumption, there is at least one task with the property from Equation (13). Assume there are only finitely many tasks with this property and let τ_j be the one of them with the highest priority. Since $s_{j,t_j} < s_{j,t_j+H}$, there exists some $t' \in [\phi_{\tau_j}, t_j]$, where τ_j is not executing at time t' but at $t' + H$. Hence, there is some higher priority task $\tau_{j'}$ which executes during t' but not during $t' + H$, i.e., $s_{j',t'} < s_{j',t'+H} = C_{j'}$ since all jobs of task $\tau_{j'}$ have the same execution time. Since $\tau_{j'}$ executes during t', we know that $\phi_{\tau_{j'}} \leq t'$. This contradicts the assumption that τ_j is the highest priority task with the property from Equation (13). □

Furthermore, similar to Leung and Whitehead [26, Lemma 3.4], we use the preceding lemma to show that the schedule repeats after $\Phi + 2H$; that is, the schedule in the interval $[\Phi + H, \Phi + 2H)$ coincides with the one in $[\Phi + 2H, \Phi + 3H)$, $[\Phi + 3H, \Phi + 4H)$, and so on. We only utilize that the total utilization $U_{\mathbb{T}} = \sum_{\tau \in \mathbb{T}} \frac{C_\tau}{T_\tau}$ of the system is at most 1, as assumed in Section 2.

Lemma 16. *When $U_{\mathbb{T}} \leq 1$, then $exec(\mathcal{S}', t) = exec(\mathcal{S}', t + H)$ holds for all $t \geq \Phi + H$.*

Proof. We assume that there is some $t \geq \Phi + H$ with $exec(\mathcal{S}', t) \neq exec(\mathcal{S}', t + H)$. Then, by Lemma 15, there is some index j with $s_{j,t} > s_{j,t+H}$. There are two cases. Either, (a) the ECU idles at some time instant $t' \in [t, t + H]$, or (b) the ECU is busy during the interval $[t, t + H]$.

For (a), by Lemma 15, $exec(\mathcal{S}', t') = (C_1, \ldots, C_n) \leq exec(\mathcal{S}', t' - H)$, i.e., the ECU also idles at time $t' - H$. Since the job releases are the same, the schedule coincides in the intervals $[t' - H, t]$ and $[t', t + H]$. Hence, $exec(\mathcal{S}', t) = exec(\mathcal{S}', t + H)$.

For (b), since $s_{j,t} > s_{j,t+H}$ and $s_{i,t} \geq s_{i,t+H}$ for all i, by Lemma 15, there is more remaining workload by jobs in the ready queue at time t than at time $t + H$. We conclude that there was more workload released during $(t, t + H]$ than could be executed by the ECU. Since the ECU did not idle between t and $t + H$, this means that $\sum_{i=1}^{n} C_i \frac{H}{T_{\tau_i}} > H$, which contradicts $\sum_{i=1}^{n} \frac{C_{\tau_i}}{T_{\tau_i}} \leq 1$. □

Based on Lemma 16, the schedule repeats after $\Phi + 2H$. More precisely, the starting time and finishing time of any mth job $J = \tau(m, jc_{max})$ of $\tau \in \mathbb{T}$ for the schedule obtained by the job collection jc_{max} that is released not before $\Phi + 2H$ can be recursively computed using the formulas:

$$-s_J^{\mathcal{S}(jc_{max})} = s_{\tau(m-\frac{H}{T_k}, jc_{max})}^{\mathcal{S}(jc_{max})} + H$$
$$-f_J^{\mathcal{S}(jc_{max})} = f_{\tau(m-\frac{H}{T_k}, jc_{max})}^{\mathcal{S}(jc_{max})} + H$$

Similarly, the starting time and finishing time of any mth job $J = \tau(m, jc_{min})$ of $\tau \in \mathbb{T}$ for the schedule obtained by the job collection jc_{min} that is released not before $\Phi + 2H$ can be recursively computed using the formulas:

$$-s_J^{\mathcal{S}(jc_{min})} = s_{\tau(m-\frac{H}{T_k}, jc_{min})}^{\mathcal{S}(jc_{min})} + H$$
$$-f_J^{\mathcal{S}(jc_{min})} = f_{\tau(m-\frac{H}{T_k}, jc_{min})}^{\mathcal{S}(jc_{min})} + H$$

As a result, for our settings with jc_{min} and jc_{max} it is sufficient to generate a schedule until all jobs that are released before $\Phi + 2H$ are finished.

In particular, the results from the previous propositions and the fact that read- and write-events are set to the starting time and finishing time enables us to provide upper and lower bounds for the read- and write-events. Independent from the job collection, they are formulated by functions $re^{min}, re^{max}, we^{min}, we^{max} : \mathbb{T} \times \mathbb{N} \to \mathbb{R}$ as follows:

Definition 17 ($re^{min}, re^{max}, we^{min}, we^{max}$). For a task set $\mathbb{T}$, we denote by $re^{min}, re^{max}, we^{min}$, we^{max} functions $\mathbb{T} \times \mathbb{N} \to \mathbb{R}$. Let $\tau \in \mathbb{T}$ and $m \in \mathbb{N}$. We define:

$$-re^{min}(\tau, m) = s_{\tau(m, jc_{min})}^{\mathcal{S}(jc_{min})} \text{ and } re^{max}(\tau, m) = s_{\tau(m, jc_{max})}^{\mathcal{S}(jc_{max})}$$
$$-we^{min}(\tau, m) = f_{\tau(m, jc_{min})}^{\mathcal{S}(jc_{min})} \text{ and } we^{max}(\tau, m) = f_{\tau(m, jc_{max})}^{\mathcal{S}(jc_{max})}$$

The values of the functions from Definition 17 at any $(\tau, m) \in \mathbb{T} \times \mathbb{N}$ are constructed as follows:

(1) We conduct two fixed-priority preemptive schedules, one for jc_{min} and one for jc_{max}, until all jobs released before $\Phi + 2H$ are finished.
(2) If $\phi_\tau + m \cdot T_\tau < \Phi + 2H$, then the values of starting time and finishing time are directly taken from the corresponding schedules.
(3) If $\phi_\tau + m \cdot T_\tau \geq \Phi + 2H$, then we calculate the read- and write-events recursively as described after Lemma 16 and set the values accordingly.

By the results from Propositions 13 and 14, we obtain that

$$-re^{min}(\tau, m) \leq \mathrm{re}_{\tau(m, jc)}^{\mathcal{S}(jc)} \leq re^{max}(\tau, m) \text{ and}$$
$$-we^{min}(\tau, m) \leq \mathrm{we}_{\tau(m, jc)}^{\mathcal{S}(jc)} \leq we^{max}(\tau, m)$$

for all tasks $\tau \in \mathbb{T}$, for all job collections $jc \in JC(\mathbb{T})$, and for all $m \in \mathbb{N}$.

5.1.2 Step 2: Bound Length of Augmented Job Chains. The fundamental step for the computation of MRT and MDA is the construction of immediate forward and immediate backward augmented job chains, respectively. However, the jobs in those job chains are associated with a job collection whereas our estimation targets to provide a bound independent of the job collection. Therefore, we need to replace the jobs in the chain with an integer indicating the job. To that end, we construct *abstract integer representations* $\vec{I}_i^E$ and $\overleftarrow{I}_i^E$, which estimate the immediate forward and immediate backward augmented job chains independent of the job chain under analysis.

An *abstract integer representation* $I \in \mathbb{N}^{|E|+2} = (i_0, \ldots, i_{|E|+1})$ is an $|E| + 2$ tuple of natural numbers. Its purpose is to define an augmented job chain $(z, J_1, \ldots, J_{|E|}, z')$ under any job collection. In particular, z is always at the read-event of the i_0th job of $E(1)$, z' is at the write-event of the $i_{|E|+1}$th job of $E(|E|)$ and J_j is the i_jth job of $E(j)$. More precisely, we say that an abstract integer representation can be evaluated at a job collection jc by applying an evaluation function.

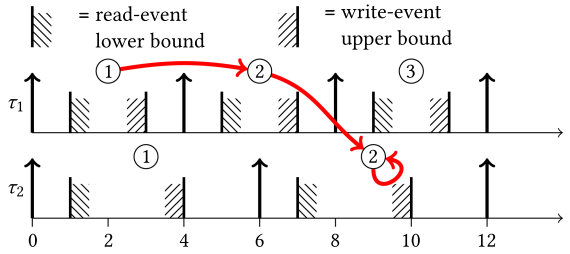


Fig. 8. Construction of $\vec{I}_1^E = (1, 2, 2, 2)$ for $E = (\tau_1 \rightarrow \tau_2)$. We assume that the read-event lower bound and write-event upper bound for each job are given and that τ_2 has higher priority than τ_1.

Definition 18. For a given cause-effect chain E we define the evaluation function

$$Eval_E : \left\{ \begin{array}{c} \text{abstract integer} \\ \text{representations} \\ \text{for } E \end{array} \right\} \times JC \rightarrow \left\{ \begin{array}{c} \text{augmented} \\ \text{job chains} \end{array} \right\}$$
$$(I = (i_0, \ldots, i_{|E|+1}), jc) \mapsto Eval_E(I, jc),$$

where $Eval_E(I, jc) = (z, J_1, \ldots, J_{|E|}, z')$ is the augmented job chain with $z = \text{re}^{\mathcal{S}(jc)}_{E(1)(i_0,jc)}$, $z' = \text{we}^{\mathcal{S}(jc)}_{E(|E|)(i_{|E|+1})}$, and $J_j = E(i)(i_j, jc)$ for all $j \in \{1, \ldots, |E|\}$.

In the following, we utilize the minimal and maximal read- and write-events determined in Section 5.1.1 to construct $\vec{I}_i^E$ and $\overleftarrow{I}_i^E$. Subsequently, we show that the evaluations $Eval_E(\vec{I}_i^E, jc)$ and $Eval_E(\overleftarrow{I}_i^E, jc)$ are augmented job chains which are not smaller than $\vec{c}_i^{E,\mathcal{S}(jc)}$ and $\overleftarrow{c}_i^{E,\mathcal{S}(jc)}$, respectively. Therefore, the evaluations can be utilized to bind the MDA and MRT.

Definition 19 (Construction of $\vec{I}_i^E$). Let $i \in \mathbb{N}$ be a natural number. The abstract integer representation $\vec{I}_i^E = (i_0, \ldots, i_{|E|+1}) \in \mathbb{N}^{|E|+2}$ is an $(|E| + 2)$-tuple with

— $i_0 = i$ and $i_1 = i + 1$.
— For $j \in \{2, \ldots, |E|\}$ the entry i_j is the smallest number in $\mathbb{N}$ such that

(1) $re^{min}(E(j), i_j) \geq we^{max}(E(j-1), i_{j-1})$, or
(2) $re^{min}(E(j), i_j)$ is no less than the i_{j-1}th release of $E(j-1)$ and $E(j-1)$ has higher priority than $E(j)$.

— $i_{|E|+1} = i_{|E|}$.

We define by $\ell(\vec{I}_i^E) := we^{max}(E(|E|), i_{|E|+1}) - re^{min}(E(1), i_0)$ the length of the abstract integer representation $\vec{I}_i^E$.

Please note that i_1 represents the $(i+1)$-th job of $E(1)$ and $i_2, \ldots, i_{|E|}$ are chosen such that the i_j-th job of $E(j)$ safely reads data after it was written by the i_{j-1}th job of $E(j-1)$ for all $j = 2, \ldots, |E|$. Furthermore, i_0 and $i_{|E|+1}$ represent the read-event of the ith job of $E(1)$ and the write-event of the $i_{|E|}$th job of $E(|E|)$, respectively. This description is similar to the description of immediate forward augmented job chains presented in Definition 5.

Example 20. Figure 8 shows how the abstract representation $\vec{I}_1^E = (1, 2, 2, 2)$ is constructed. The first entry is set to 1 due to the index of $\vec{I}_1^E$. The second entry is computed by $1 + 1$, which represents the second job of τ_1. The write-event upper bound of the second job of τ_1 is lower than the read-event of the second job of τ_2 under *any* job collection, since the write-event upper bound

$we^{max}(\tau_1, 2) = 7$, is no less than the read-event lower bound $re^{max}(\tau_2, 2) = 7$. The last value of $\vec{I}_1^E = (1, 2, 2, 2)$ is simply repeated to account for the event for actuation.

The abstract integer representation yields an upper bound on the immediate forward augmented job chains as stated in the following lemma.

Lemma 21 (Forward Bound). *Let $\vec{I}_i^E = (i_0, \ldots, i_{|E|+1})$ be some abstract representation as specified in Definition 19. We have the following inequality*

$$\ell\left(\vec{c}_i^{E,S(jc)}\right) \leq \ell\left(Eval_E\left(\vec{I}_i^E, jc\right)\right) \leq \ell\left(\vec{I}_i^E\right), \tag{14}$$

for all job collections $jc \in JC(\mathbb{T})$.

Proof. We first show that $\ell(\vec{c}_i^{E,jc}) \leq \ell(Eval_E(\vec{I}_i^E, jc))$. In the following, we use the notation $\vec{c}_i^{E,jc} = (z, J_1, \ldots, J_{|E|}, z')$ and $Eval_E(\vec{I}_i^E, jc) = (\tilde{z}, \tilde{J}_1, \ldots, \tilde{J}_{|E|}, \tilde{z}')$. Since the first entry of $\vec{I}_i^E$ is i, we know that $\tilde{z}$ is at the read-event of the ith job of $E(1)$. In addition, z is at the read-event of the ith job of $E(1)$. Therefore, $z = \tilde{z}$. Moreover, J_1 and $\tilde{J}_1$ are both the $(i+1)$-th job of $E(1)$ and, therefore, coincide as well. In the following we show that

$$r_{J_j} \leq r_{\tilde{J}_j}, \tag{15}$$

for all $j \in \{1, \ldots, |E|\}$ by induction. For $j = 1$, Equation (15) holds by the above discussion. Assume Equation (15) holds for $j-1$, then the write-event of J_{j-1} is no later than the write-event of $\tilde{J}_{j-1}$. By Definition 19, the job $\tilde{J}_j$ reads data after it was written by $\tilde{J}_{j-1}$. Therefore, $\tilde{J}_j$ reads data after it was written by J_{j-1} as well. Since J_j is the earliest job which reads data that was written by J_{j-1}, it must be released not later than $\tilde{J}_j$, i.e., $r_{J_j} \leq r_{\tilde{J}_j}$. This concludes the proof of Equation (15) by induction. Since $r_{J_{|E|}} \leq r_{\tilde{J}_{|E|}}$, $J_{|E|}$ writes data no earlier than $\tilde{J}_{|E|}$ writes data, and $\tilde{z}' \geq z'$. We conclude that $\ell(\vec{c}_i^{E,S(jc)}) = (z' - z) \leq (\tilde{z}' - \tilde{z}) = \ell(Eval_E(\vec{I}_i^E, jc))$.

Next, we prove $\ell(Eval_E(\vec{I}_i^E, jc)) \leq \ell(\vec{I}_i^E)$. Let $\vec{I}_i^E = (i_0, \ldots, i_{|E|+1})$. By Propositions 13 and 14, $re^{min}(E(1), i_0) \leq \mathrm{re}_{E(1)(i_0,jc)} = z$ and $we^{max}(E(|E|), i_{|E|+1}) \geq \mathrm{we}_{E(|E|)(i_{|E|+1},jc)} = z'$ hold. We conclude that $\ell(Eval_E(\vec{I}_i^E, jc)) = z' - z \leq we^{max}(E(|E|), i_{|E|+1}) - re^{min}(E(1), i_0) = \ell(\vec{I}_i^E)$. □

Now, we consider $\overleftarrow{I}_i^E$, constructed as follows.

Definition 22 (Construction of $\overleftarrow{I}_i^E$). Let $i \in \mathbb{N}$ be a natural number. The abstract integer representation $\overleftarrow{I}_i^E = (i_0, \ldots, i_{|E|+1}) \in \mathbb{N}^{|E|+2}$ is a $(|E|+2)$-tuple with:

- $i_{|E|+1} = i$ and $i_{|E|} = i - 1$.
- For $j \in \{|E|-1, \ldots, 1\}$ the entry i_j is the highest number in $\mathbb{N}$ such that
  (1) $re^{min}(E(j+1), i_{j+1}) \geq we^{max}(E(j), i_j)$, or
  (2) $E(j)$ has higher priority than $E(j+1)$ and $re^{min}(E(j+1), i_{j+1})$ is no earlier than the release of the i_jth job of $E(j)$.
- $i_0 = i_1$.

We define by $\ell(\overleftarrow{I}_i^E) := we^{max}(E(|E|), i_{|E|+1}) - re^{min}(E(1), i_0)$ the length of the abstract integer representation $\overleftarrow{I}_i^E$.

As for immediate backward augmented job chains, we call the abstract representation $\overleftarrow{I}_i^E$ *complete*, if it can be fully constructed. Otherwise, we call it *incomplete*. Note that this construction of $\overleftarrow{I}_i^E$ is similar to the construction of immediate backward augmented job chains in Definition 6.

By using the evaluation function specified in Definition 18, we provide a bound for the immediate backward augmented job chains as well. The only difference with respect to Lemma 21 is that we have to exclude the incomplete representations.

Lemma 23 (Backward Bound). *Let $\overleftarrow{I}_i^E = (i_0, \ldots, i_{|E|+1})$ be some complete abstract representation. Then $\overleftarrow{c}_i^{E,jc}$ is complete as well and we have the following inequality*

$$\ell\left(\overleftarrow{c}_i^{E,\mathcal{S}(jc)}\right) \leq \ell\left(Eval_E\left(\overleftarrow{I}_i^E, jc\right)\right) \leq \ell\left(\overleftarrow{I}_i^E\right), \tag{16}$$

for all job collections jc of $\mathbb{T}$.

Proof. Let $\overleftarrow{I}_i^E = (i_0, \ldots, i_{|E|+1})$ be complete. Moreover, let $\overleftarrow{c}_i^{E,\mathcal{S}(jc)} = (z, J_1, \ldots, J_{|E|}, z')$ and $Eval_E(\overleftarrow{I}_i^E, jc) = (\tilde{z}, \tilde{J}_1, \ldots, \tilde{J}_{|E|}, \tilde{z}')$. We have $z' = \tilde{z}'$ and $J_{|E|} = \tilde{J}_{|E|}$. Analogous to the proof of Lemma 21, we obtain $r_{J_j} \geq r_{\tilde{J}_j}$ for all $j \in \{|E|, \ldots, 1\}$ inductively since we have $we_{\tilde{J}_j} \leq re_{\tilde{J}_{j+1}} \leq re_{J_{j+1}}$ and J_j is the latest job which writes before re_{J_j}. In the end, we obtain $z = re_{J_1} \geq re_{\tilde{J}_1} = \tilde{z}$. We conclude that $\ell(\overleftarrow{c}_i^{E,\mathcal{S}(jc)}) = z' - z \leq \tilde{z}' - \tilde{z} = \ell(Eval_E(\overleftarrow{I}_i^E, jc))$. Please note that since for all j a job $\tilde{J}_j$ exists which writes data before the read-event of J_{j-1}, the immediate backward job chain $\overleftarrow{c}_i^{E,\mathcal{S}(jc)}$ can be fully constructed and is therefore complete.

As in the proof of Lemma 21, we observe that $\ell(Eval_E(\overleftarrow{I}_i^E, jc)) = \tilde{z}' - \tilde{z}$ is upper bounded by $we^{max}(E(|E|), i_{|E|+1}) - re^{min}(E(1), i_0) = \ell(\overleftarrow{I}_i^E)$. This concludes the proof of Equation (16). □

However, there may still be some immediate backward augmented job chains $\overleftarrow{c}_i^{E,\mathcal{S}(jc)}$ which are complete but whose corresponding abstract representation $\overleftarrow{I}_i^E$ is not. For those chains, we provide an upper bound in the following.

Lemma 24 (Second Backward Bound). *Let $i \in \mathbb{N}$ be a natural number and let jc be some collection of jobs for $\mathbb{T}$, such that the immediate backward augmented job chain $\overleftarrow{c}_i^{E,\mathcal{S}(jc)}$ is complete. The length of $\overleftarrow{c}_i^{E,\mathcal{S}(jc)}$ is upper bounded by*

$$\ell\left(\overleftarrow{c}_i^{E,\mathcal{S}(jc)}\right) \leq we^{max}(E(|E|), i) - re^{min}(E(1), 1) =: B(i). \tag{17}$$

Proof. Let $\overleftarrow{c}_i^{E,\mathcal{S}(jc)} = (z, J_1, \ldots, J_{|E|}, z')$. Then z' is at the write-event of the ith job of $E(|E|)$, i.e., $z' \leq we^{max}(E(|E|), i)$. Moreover, if $\overleftarrow{c}_i^{E,\mathcal{S}(jc)}$ is complete, then the job J_1 of $E(1)$ exists. The read-event of that job (which equals z) is no earlier than the read-event of the first job of $E(1)$. Therefore, $z \geq re^{min}(E(1), 1)$. We conclude $\ell(\overleftarrow{c}_i^{E,\mathcal{S}(jc)}) = z' - z \leq we^{max}(E(|E|), i) - re^{min}(E(1), 1)$. □

5.1.3 Step 3: Provide End-To-End Latency. By Lemma 21 (respectively, Lemmas 23 and 24), the computation of MRT (respectively, MDA) is reduced to a construction of the abstract representations $\vec{I}_i^E$ (respectively, $\overleftarrow{I}_i^E$). As proven in Section 5.1.1, the upper and lower bounds on the read- and write-events repeat each hyperperiod after $\Phi + 2H$. Therefore, the construction and also the length of the abstract integer representations $\vec{I}_i^E$ and $\overleftarrow{I}_i^E$ repeats as well and only a finite number of them has to be considered to provide a latency bound. The following two lemmas show that it is sufficient to consider abstract integer representations where the job that is described by the first entry is released until $\Phi + 2H$.

Lemma 25. *Let $\vec{I}_i^E = (i_0, \ldots, i_{|E|+1})$ be an abstract integer representation with $\phi_{E(1)} + (i_0 - 1) \cdot T_{E(1)} \geq \Phi + 2H$. There exists $j \in \mathbb{N}$ such that for $\vec{I}_j^E = (j_0, \ldots, j_{|E|+1})$:*

$$-\phi_{E(1)} + (j_0 - 1)T_{E(1)} \in [\Phi + H, \Phi + 2H)$$
$$-\ell(\vec{I}_j^E) = \ell(\vec{I}_i^E)$$

Proof. Since $Z := [\Phi + H, \Phi + 2H)$ has length H and $w := \phi_{E(1)} + (i_0 - 1) \cdot T_{E(1)}$ is no less than the right boundary of Z, there exists some $\xi \in \mathbb{N}$ such that $w - \xi \cdot H \in Z$. We choose $j := i - \xi \cdot \frac{H}{T_{E(1)}}$ and consider $\vec{I}_j^E$. Since the maximal and minimal read- and write-events repeat each hyperperiod H, the underlying job sequence of $\vec{I}_j^E$ is just the abstract representation of $\vec{I}_i^E$ shifted ξ hyperperiods to the left. Therefore $\phi_{E(1)} + (j_0 - 1) \cdot T_{E(1)} = w - \xi \cdot H \in Z$ and $\ell(\vec{I}_j^E) = \ell(\vec{I}_i^E)$. □

Please note that since for $\vec{I}_i^E$ the first entry is $i_0 = i$, if i is increased, then i_0 is increased as well. As a consequence, to find all i such that $\phi_{E(1)} + (i_0 - 1)T_{E(1)} < \Phi + 2H$ is fulfilled, it is sufficient to consider $i = 1, 2, 3, \ldots$ and stop as soon as $\phi_{E(1)} + (i_0 - 1)T_{E(1)} \geq \Phi + 2H$.

In a similar way, we formulate the lemma for $\overleftarrow{I}_i^E$.

Lemma 26. *Let $\overleftarrow{I}_i^E = (i_0, \ldots, i_{|E|+1})$ be an abstract integer representation with $\phi_{E(1)} + (i_0 - 1) \cdot T_{E(1)} \geq \Phi + 2H$. There exists $j \in \mathbb{N}$ such that for $\overleftarrow{I}_j^E = (j_0, \ldots, j_{|E|+1})$:*

$$-\phi_{E(1)} + (j_0 - 1)T_{E(1)} \in [\Phi + H, \Phi + 2H)$$
$$-\ell\left(\overleftarrow{I}_j^E\right) = \ell\left(\overleftarrow{I}_i^E\right)$$

Proof. The proof is analogous to the proof of Lemma 25, except that $j := i - \xi \cdot \frac{H}{T_{E(|E|)}}$. □

Please note that for $\overleftarrow{I}_i^E$ the first entry is not necessarily $i_0 = i$. However, if i is increased, then i_0 is increased as well as proven in the following lemma. As a consequence, to find all i such that $\phi_{E(1)} + (i_0 - 1)T_{E(1)} < \Phi + 2H$ is fulfilled, it is sufficient to consider $i = 1, 2, 3, \ldots$ and stop as soon as $\phi_{E(1)} + (i_0 - 1)T_{E(1)} \geq \Phi + 2H$.

Lemma 27. *Let $\overleftarrow{I}_i^E = (i_0, \ldots, i_{|E|+1})$ and $\overleftarrow{I}_j^E = (j_0, \ldots, j_{|E|+1})$. If $i \leq j$, then $i_0 \leq j_0$ as well.*

Proof. It holds $i_{|E|+1} = i \leq j = j_{|E|+1}$. Now we perform induction over $k = |E|, \ldots, 1$ to show that $i_k \leq j_k$. Since i_k fulfills properties (1) and (2) in Definition 22 with respect to i_{k+1}, and $re^{min}(E(k+1), i_{k+1}) \leq re^{min}(E(k+1), j_{k+1})$ by induction, i_k also fulfills properties (1) and (2) with respect to j_{k+1}. Since j_k is the maximal value that fulfills (1) and (2) with respect to j_{k+1}, it holds $i_k \leq j_k$. We conclude $i_0 = i_1 \leq j_1 = j_0$. □

Under the assumption that the read- and write-event upper and lower bounds as defined in Definition 17 are computed for all jobs, we can construct $\vec{I}_i^E$ and $\overleftarrow{I}_i^E$ for all $i = 1, 2, 3, \ldots$ until $\phi_{E(1)} + (j_0 - 1)T_{E(1)} \geq \Phi + 2H$. The abstract representations $\vec{I}_i^E$ and $\overleftarrow{I}_i^E$ provide an upper bound on the length $\ell(\vec{c}_i^{E,\mathcal{S}(jc)})$ of the corresponding immediate forward augmented job chain $\vec{c}_i^{E,\mathcal{S}(jc)}$ and on the length $\ell(\overleftarrow{c}_i^{E,\mathcal{S}(jc)})$ of the corresponding immediate backward augmented job chains $\overleftarrow{c}_i^{E,\mathcal{S}(jc)}$ by Lemmas 21 and 23, respectively. By Lemmas 25 and 26 this finite number of abstract representations is sufficient to bound the length of *all* immediate forward/backward augmented job chains. By taking the maximum $\max_i \ell(\vec{I}_i^E)$ and $\max_i \ell(\overleftarrow{I}_i^E)$ of the length of these finitely many abstract representations, we obtain upper bounds on the maximum reaction time $\mathrm{MRT}(E)$ and maximum data age $\mathrm{MDA}(E)$, namely, Equations (7) and (8), respectively. Please note that we need to account for the complete $\overleftarrow{c}_i^{E,\mathcal{S}(jc)}$ which do not have complete $\overleftarrow{I}_i^E$ by Lemma 24.

Theorem 28 (Upper Bound for Implicit Communication). *Let E be a cause-effect chain where all tasks are on one ECU and all tasks adhere to implicit communication. We denote by $\mathcal{I}^{fw}$ the set of all $i \in \mathbb{N}$ such that $\phi_{E(1)} + (i_0 - 1)T_{E(1)} < \Phi + 2H$ for $\vec{I}_i^E = (i_0, \ldots, i_{|E|+1})$. Furthermore, we denote*

ALGORITHM 1: MRT bound under implicit communication.

```
Compute we^max and re^min by conducting concrete schedules.        ▷ After Definition 17
I^fw = {}, i_0 = 1, i = 1
while True do
    Construct I⃗_i.                                                 ▷ Definition 19
    i_0 := first entry of I⃗_i
    if φ_E(1) + (i_0 − 1)T_E(1) < Φ + 2H then
        Add i to I^fw.
        i = i + 1
    else
        break
Compute upper bound from Equation (18).
```

ALGORITHM 2: MDA bound under implicit communication.

```
Compute we^max and re^min by conducting concrete schedules.        ▷ After Definition 17
I^bw = {}, i_0 = 1, i = 2
while True do
    Construct I⃖_i.                                                 ▷ Definition 22
    i_0 := first entry of I⃖_i
    if φ_E(1) + (i_0 − 1)T_E(1) < Φ + 2H then
        Add i to I^bw.
        i = i + 1
    else
        break
Compute upper bound from Equation (19).
```

by $\mathcal{I}^{bw}$ the set of all $i \in \mathbb{N} - \{1\}$ such that $\phi_{E(1)} + (i_0 - 1)T_{E(1)} < \Phi + 2H$ for $\overleftarrow{I}_i^E = (i_0, \ldots, i_{|E|+1})$. The following equations hold:

$$\mathrm{MRT}(E) \leq \max_{i \in \mathcal{I}^{fw}} \ell\left(\vec{I}_i^E\right), \tag{18}$$

$$\mathrm{MDA}(E) \leq \max_{i \in \mathcal{I}^{bw}} \begin{cases} \ell\left(\overleftarrow{I}_i^E\right), & \overleftarrow{I}_i^E \textit{ complete} \\ B(i), & \textit{otherwise.} \end{cases} \tag{19}$$

Proof. According to Equation (7), the MRT of the cause-effect chain E is defined by $\sup_{i\in\mathbb{N}} \sup_{jc\in JC} \ell(\vec{c}_i^{E,\mathcal{S}(jc)})$. Since $\ell(\vec{c}_i^{E,\mathcal{S}(jc)}) \leq \ell(\vec{I}_i^E)$ for all $jc \in JC$ by Lemma 21, the MRT is upper bounded by $\sup_{i\in\mathbb{N}} \sup_{jc\in JC} \ell(\vec{I}_i^E)$. Since $\ell(\vec{I}_i^E)$ is independent from jc, we obtain the upper bound $\sup_{i\in\mathbb{N}} \ell(\vec{I}_i^E)$. We conclude that by Lemma 25 only those abstract representations with $i \in \mathcal{I}^{fw}$ have to be considered. This yields Equation (18).

By Equation (8), the MDA of E is $\sup_{i\in\mathbb{N}} \sup_{jc\in JC}\{\ell(\overleftarrow{c}_i^{E,\mathcal{S}(jc)}) | \overleftarrow{c}_i^{E,\mathcal{S}(jc)} \text{ complete}\}$. If $\overleftarrow{I}_i^E$ is complete, then $\overleftarrow{c}_i^{E,\mathcal{S}(jc)}$ is complete and we know that $\ell(\overleftarrow{c}_i^{E,\mathcal{S}(jc)}) \leq \ell(\overleftarrow{I}_i^E)$ by Lemma 23. If $\overleftarrow{I}_i^E$ is *not* complete but $\overleftarrow{c}_i^{E,\mathcal{S}(jc)}$ is complete, then we know $\ell(\overleftarrow{c}_i^{E_p,\mathcal{S}(jc)}) \leq B(i)$ by Lemma 24. Hence, we conclude that Equation (19) holds. □

The procedures to compute our MRT and MDA bounds are shown in Algorithms 1 and 2, respectively. Please note that although the MDA is bounded by the MRT, the bounds provided in

this section are over-approximations and do not necessarily follow this ordering. In particular, the bound obtained for the MDA may be larger than the bound for the MRT.

Furthermore, we note that in the case that $C_\tau^u = C_\tau^\ell$ for all tasks $\tau \in \mathbb{T}$, the execution time of every job of each task is fixed. In this case, only the job collection $jc = jc_{max} = jc_{min}$ must be considered and the maximal and minimal read- and write-events coincide with the actual read- and write-events. As a result, the abstract integer representations $\vec{I}_i^E$ and $\overleftarrow{I}_i^E$ constructed in Definitions 19 and 22 always evaluate to $\vec{c}_i^{E,\mathcal{S}(jc)}$ and $\overleftarrow{c}_i^{E,\mathcal{S}(jc)}$, respectively. Hence, the MDA and the MRT upper bounds obtained by Theorem 28 when excluding $B(i)$ from Equation (19) are exact.

Corollary 29. *If $C_\tau^u = C_\tau^\ell$ holds for all tasks $\tau \in \mathbb{T}$, then* $\mathrm{MRT}(E) = \max_{i \in \mathcal{I}^{fw}} \ell(\vec{I}_i^E)$ *and* $\mathrm{MDA}(E) = \max_{i \in \mathcal{I}^{bw}} \ell(\overleftarrow{I}_i^E)$ *hold as well.*

Proof. If $C_\tau^u = C_\tau^\ell$ for all tasks $\tau \in \mathbb{T}$, then there is only one job collection in $JC(\mathbb{T}) = \{jc\}$ and one schedule $\mathcal{S}(jc)$. Moreover, the maximal and minimal read- and write-events utilized in the construction of $\vec{I}_i^E$ and $\overleftarrow{I}_i^E$ are the actual read- and write-events of the jobs, respectively. Hence, $\ell(\vec{c}_i^{E,\mathcal{S}(jc)}) = \ell(Eval_E(\vec{I}_i^E, jc)) = \ell(\vec{I}_i^E)$. Moreover, $\overleftarrow{c}^{E,\mathcal{S}(jc)}$ is complete if and only if $\overleftarrow{I}_i^E$ is complete and we have $\ell(\overleftarrow{c}_i^{E,\mathcal{S}(jc)}) = \ell(Eval_E(\overleftarrow{I}_i^E, jc)) = \ell(\overleftarrow{I}_i^E)$. We conclude that $\mathrm{MRT}(E) = \sup_i \ell(\vec{c}_i^{E,\mathcal{S}(jc)}) = \sup_i \ell(\vec{I}_i^E) = \max_{i \in \mathcal{I}^{fw}} \ell(\vec{I}_i^E)$ and $\mathrm{MDA}(E) = \sup_i \ell(\overleftarrow{c}_i^{E,\mathcal{S}(jc)}) = \sup_i \ell(\overleftarrow{I}_i^E) = \max_{i \in \mathcal{I}^{bw}} \ell(\overleftarrow{I}_i^E)$. □

Complexity: Two components play a decisive role for the time complexity of our analysis. We (a) create the schedule for the job collections jc_{max} and jc_{min} for the bounded time frame to obtain minimal and maximal read- and write-events for each job, and (b) create and compare abstract integer representations based on the minimal and maximal read- and write-events. We examine the time complexity for a cause-effect chain E on an ECU with task set $\mathbb{T} = \{\tau_1, \ldots, \tau_n\}$ with n tasks.

Since the schedule repeats after $\Phi + 2H$, it suffices to schedule the jobs in the interval $[0, \Phi + 2H)$. Hence, the time complexity for (a) is $O(\frac{\Phi+2H}{T_{min}} \cdot n)$, where $T_{min} := \min_{\tau \in \mathbb{T}} T_\tau$ is the minimal period in $\mathbb{T}$.

For each abstract integer representation we have to determine $|E| + 2$ integers. There is a cost of query[1] Q depending on the data structure for finding the next integer for the abstract representation. To compute the MRT, we have to simulate and compare up to $\frac{\Phi+2H}{T_{E(1)}}$ abstract integer representations, and for the MDA up to $\frac{\Phi+2H}{T_{E(|E|)}}$ abstract integer representations are under analysis. Hence, the time complexity for component (b) is $O(|E| \cdot Q \cdot \frac{\Phi+2H}{T_{E(1)}})$ for reaction time and $O(|E| \cdot Q \cdot \frac{\Phi+2H}{T_{E(|E|)}})$ for data age.

We note that the time complexity for the method by Kloda et al. [22] coincides with the complexity of component (b) for our reaction time computation, except that they have to call a latency function for each job instead of determining the job itself. The methods by Dürr et al. [11] and by Davare [10] have complexity $O(|E|)$. Since these methods all assume that the WCRTs are known, i.e., computed in advance, the time complexity of the WCRT computation should also be taken into account.

5.2 Interconnected Analysis

In this subsection, we analyze the timing behavior of cause-effect chains that are distributed among several ECUs. If clock shifts are known and all tasks (even communication tasks) behave like

[1]The data structure can be designed such that $Q = O(1)$. Let all jobs for each task τ be stored in a list l_τ, ordered by their release. If we want to find the first job of τ with minimal read-event after a time instant t and assume that all jobs finish their execution before the subsequent job release, then only those jobs in the list with index j between $\lceil \frac{t-\phi_\tau-T_\tau}{T_\tau} \rceil$ and $\lfloor \frac{t-\phi_\tau}{T_\tau} \rfloor$ are candidates to be checked, i.e., $O(T_\tau / T_\tau)$ many jobs.

periodic tasks, then the analysis from the preceding subsection can be utilized. However, since global clock synchronization is often avoided in distributed real-time systems to reduce failure dependencies, the clock shifts between different ECUs are unknown by the observer. Moreover, the implementation of communication tasks varies depending on the underlying architecture, e.g., these tasks may behave in a non-preemptive or non-periodic manner. This hinders the exact determination of data age and reaction time.

We discuss how to provide proper upper bounds on data age and reaction time of interconnected cause-effect chains. Our estimations utilize only knowledge about the WCRT R_{τ^c} and maximum inter-arrival time $T^{max}_{\tau^c}$, i.e., maximum time between two recurrent job releases, of each communication task τ^c.

Let $\mathcal{S}$ be a schedule of the task set $\mathbb{T}$. Consider some interconnected cause-effect chain IE of $\mathbb{T}$. We separate IE into local cause-effect chains $E_1, \dots, E_k$ with communication tasks $\tau^c_1, \tau^c_2, \dots, \tau^c_{k-1}$. More specifically, we have

$$IE = (E_1 \to \tau^c_1 \to E_2 \to \tau^c_2 \to \cdots \to \tau^c_{k-1} \to E_k), \tag{20}$$

where each $E_i, i = 1, \dots, k$ only contains tasks on a single ECU, namely, $ECU(E_i)$, and each $\tau^c_i, i = 1, \dots, k-1$ communicates from $ECU(E_i)$ to $ECU(E_{i+1})$.

We utilize the Cutting-Theorem (Theorem 12) to estimate the reaction time and data age of IE by its local components, i.e., $\mathrm{MRT}(IE, \mathcal{S}) \le \sum_{i=1}^{k} \mathrm{MRT}(E_i, \mathcal{S}) + \sum_{i=1}^{k-1} \mathrm{MRT}((\tau^c_i), \mathcal{S})$ and $\mathrm{MDA}(IE, \mathcal{S}) \le \sum_{i=1}^{k} \mathrm{MDA}(E_i, \mathcal{S}) + \sum_{i=1}^{k-1} \mathrm{MDA}((\tau^c_i), \mathcal{S})$. Please note that $(\tau^c_i), i = 1, \dots, k-1$ can be considered as a cause-effect chain of just one task. We apply the bound from Davare et al. [10] to estimate data age and reaction time of (τ^c_i) by $T^{max}_{\tau^c_i} + R_{\tau^c_i}$ under implicit communication.

Corollary 30. *The MRT and data age of the interconnected cause-effect chain IE under implicit communication can be estimated by the timing behavior of its local parts:*

$$\mathrm{MRT}(IE, \mathcal{S}) \le \sum_{i=1}^{k} \mathrm{MRT}(E_i, \mathcal{S}) + \sum_{i=1}^{k-1} (T^{max}_{\tau^c_i} + R_{\tau^c_i}), \tag{21}$$

$$\mathrm{MDA}(IE, \mathcal{S}) \le \sum_{i=1}^{k} \mathrm{MDA}(E_i, \mathcal{S}) + \sum_{i=1}^{k-1} (T^{max}_{\tau^c_i} + R_{\tau^c_i}). \tag{22}$$

Proof. The result follows from the Cutting-Theorem (Theorem 12) together with the estimation by Davare et al. [10] as discussed above. □

We note that the values of $\mathrm{MRT}(E_i, \mathcal{S})$ and $\mathrm{MDA}(E_i, \mathcal{S})$ can be upper bounded by applying our Analysis in Section 5.1.

Under LET we obtain a similar result.

Corollary 31. *The MRT and data age of the interconnected cause-effect chain IE under LET can be estimated by the timing behavior of its local parts:*

$$\mathrm{MRT}(IE, \mathcal{S}) \le \sum_{i=1}^{k} \mathrm{MRT}(E_i, \mathcal{S}) + \sum_{i=1}^{k-1} 2T^{max}_{\tau^c_i}, \tag{23}$$

$$\mathrm{MDA}(IE, \mathcal{S}) \le \sum_{i=1}^{k} \mathrm{MDA}(E_i, \mathcal{S}) + \sum_{i=1}^{k-1} 2T^{max}_{\tau^c_i}. \tag{24}$$

Proof. This also follows from the Cutting-Theorem (Theorem 12). We use that under LET $\mathrm{MDA}((\tau^c_i), \mathcal{S}) \le \mathrm{MRT}((\tau^c_i), \mathcal{S}) \le 2T^{max}_{\tau^c_i}$. □

For Corollary 31, the values of $\mathrm{MRT}(E_i, \mathcal{S})$ and $\mathrm{MDA}(E_i, \mathcal{S})$ can be computed by comparing valid augmented job chains with events for external activity during $[0, \Phi + H)$.

Please note that a similar approach can be utilized to analyze end-to-end latencies on multi-processor systems with partitioned scheduling algorithms, even if the cores are not synchronized. In such a case each subchain E_i represents a part of the cause-effect chain E that is executed on one processor (instead of one ECU) and the communication tasks τ_i^c account for the core communication.

6 ALTERNATIVE DATA AGE DEFINITION

The definition of MDA from other analyses [4, 11, 22] differs from our definition of MDA in the following way: If we consider the chain of events as outlined in Figure 3, their MDA includes only the time from sampling until the processed-event and not until the actuation-event. This does not mean that their estimation is more precise, but that they bound a smaller time interval. For comparison with the MDA from [4, 11, 22], we introduce the definition of a *maximum reduced data age* $\mathrm{MRDA}(E, \mathcal{S})$. It follows Definition 6, except that we set the event for actuation to the processed-event.

Definition 32 (Reduced Immediate Backward Augmented Job Chain). A *reduced immediate backward augmented job chain* $\overleftarrow{c}^{*E,\mathcal{S}}_m$ with $m \in \mathbb{N}$ is the unique augmented job chain $(z, J_1, \ldots, J_{|E|}, z')$ where

- the actuation is set to the time of the processed-event, which happens at the mth write-event of $E(|E|)$, i.e., $z' = we_{J_{|E|}}$ and $J_{|E|} = E(|E|)(m)$,
- the sequence $(J_1, \ldots, J_E)$ is an immediate backward job chain for E in $\mathcal{S}$, and
- the external activity is set to the time where the data is sampled, i.e., $z = re_{J_1}$.

More specifically, the first $|E| + 1$ entries of $\overleftarrow{c}^{*E,\mathcal{S}}_m$ coincide with those of $\overleftarrow{c}^{E,\mathcal{S}}_{m+1}$ and z' is set to $we_{J_{|E|}}$. We define $\overleftarrow{c}^{*E,\mathcal{S}}_m$ to be *valid* if and only if $\overleftarrow{c}^{E,\mathcal{S}}_{m+1}$ is valid in the sense of Definition 10. The reduced data age is defined similar to Definition 11:

Definition 33 (Maximum Reduced Data Age). For a cause-effect chain E with schedule $\mathcal{S}$ we define the *schedule specific maximum reduced data age* by

$$\mathrm{MRDA}(E, \mathcal{S}) := \sup \left\{ \ell\left(\overleftarrow{c}^{*E,\mathcal{S}}_m\right) \,\middle|\, m \in \mathbb{N}, \overleftarrow{c}^{*E,\mathcal{S}}_m \text{ valid} \right\}, \tag{25}$$

where ℓ is the length of an augmented job chain as defined in Equation (1). As before, the *overall maximum reduced data age* is obtained by the supremum over all schedules, i.e., $\mathrm{MRDA}(E) = \sup_{\mathcal{S}} \mathrm{MRDA}(E, \mathcal{S})$.

The Cutting-Theorem (Theorem 12) is transferred to the defined reduced data age as follows. In the proof, we cut off an immediate backward augmented job chain at the beginning. This works independently from the choice of z', i.e., we cut off an immediate backward augmented job chain also from a reduced immediate backward augmented job chain. For $E = (E_1 \to E_2)$ this leads to

$$\mathrm{MRDA}(E, \mathcal{S}) \leq \mathrm{MDA}(E_1, \mathcal{S}) + \mathrm{MRDA}(E_2, \mathcal{S}). \tag{26}$$

The computation in the local case is similar to Section 5.1. However, to match the pattern of $\overleftarrow{c}^{*E,\mathcal{S}}_i$ we introduce the abstract integer representation $\overleftarrow{I}^{*E}_i$ which are constructed by the same rule set as $\overleftarrow{I}^{E}_i$ but the last *two* entries are set to i. With periodic job releases and fixed execution times, the schedule repeats after $\Phi + 2H$ where Φ is the maximal phase and H is the hyperperiod of tasks on that ECU, respectively. Therefore, it suffices to construct and compare all $\overleftarrow{I}^{*E}_i$ such that the first entry of $\overleftarrow{I}^{*E}_i$ is less than $\Phi + 2H$, i.e., we denote this set of i by $\mathcal{I}^{bw*}$. We obtain the bound:

$$\mathrm{MRDA}(E) \leq \max_{i \in \mathcal{I}^{bw*}} \begin{cases} \ell\left(\overleftarrow{I^{*E}_i}\right), & \overleftarrow{I^{*E}_i} \text{ complete} \\ B(i), & else \end{cases}. \tag{27}$$

For the interconnected case, we rely on the new cutting theorem from Equation (26) and obtain

$$\mathrm{MRDA}(IE, \mathcal{S}) \leq \sum_{i=1}^{k-1} (\mathrm{MDA}(E_i, \mathcal{S}) + T^{max}_{\tau^c_i} + R_{\tau^c_i}) + \mathrm{MRDA}(E_k, \mathcal{S}), \tag{28}$$

for any interconnected cause-effect chain $IE = (E_1 \rightarrow \tau^c_1 \rightarrow E_2 \rightarrow \cdots \rightarrow \tau^c_{k-1} \rightarrow E_k)$ as in Corollary 30. The local values of maximum (respectively, maximum reduced) data age from Equation (28) are computed by simulating all the abstract integer representations $\overleftarrow{I^E_i}$ (respectively, $\overleftarrow{I^{*E}_i}$) in a bounded time frame and using Equation (19) (respectively, Equation (27)).

7 EVALUATION

A relevant industrial use-case of the presented end-to-end latency analyses is the timing verification of cause-effect chains in the automotive domain. To assess the practical benefit of our proposed analyses, we evaluated it using synthesized task sets and cause-effect chains that adhere to the details described in *Automotive Benchmarks For Free* [25]. Furthermore, we generated task sets with the UUnifast algorithm [7] to assess the performance for general task parameters. We consider periodic task sets and implicit job communication to apply our analysis results from Sections 5 and 6. Two setups are considered in the evaluation, which is released on Github [30].

Intra-ECU setup: All tasks in each cause-effect chain are mapped to one ECU and have a locally synchronized clock.

Inter-ECU setup: Tasks within a cause-effect chain are mapped to different ECUs that are not synchronized. An interconnect fabric is used for data communication across different ECUs.

In the following, we use the method by Davare et al. [10] to normalize all other end-to-end bounds, since this method yields the most pessimistic result. We define the *latency reduction* $LR(\mathcal{M})$ of an analysis method $\mathcal{M}$ with respect to an evaluated bound $B(\cdot)$, e.g., MRT, by

$$LR(\mathcal{M}) := (B(Davare) - B(\mathcal{M}))/B(Davare). \tag{29}$$

Additionally, for the Intra-ECU case where all jobs execute the WCET (i.e., $BCET = WCET$), an exact analysis of MRT and MDA is given by Corollary 29. Since this is a valid execution scenario for the case where $BCET \leq WCET$, Corollary 29 provides a lower bound for the case $BCET \leq WCET$. Therefore, we also consider the *gap reduction* $GR(\mathcal{M})$, defined by

$$GR(\mathcal{M}) := (B(Davare) - B(\mathcal{M}))/(B(Davare) - B(\text{Corollary 29})). \tag{30}$$

In particular, the closer the gap reduction is to 1, the tighter that bound is to the lower bound, and, therefore, the tighter that value is to the exact latency bound.

7.1 Task and Task Set Generation

In this evaluation, a task τ_i is described by the WCET C_i, period T_i, phase ϕ_i, and priority π_i. Furthermore, $U_i = C_i/T_i$ is the utilization of task τ_i.

Automotive benchmark [25]: A task τ_i is generated as follows[2]:

[2]In the automotive benchmark [25], atomic software components contain *runnables* subject to scheduling. Multiple runnables with the same period are afterward grouped together as a task. Since communication usually happens on the runnable level, we set up the experiments accordingly, and denote each runnable as a task to match the common notation in real-time systems research and the cause-effect chain model in the literature.

(1) The period T_i in *ms* of a task τ_i is drawn from the set $T = \{1, 2, 5, 10, 20, 50, 100, 200, 1{,}000\}$ according to the related share[3] of [25, Table III, IV, and V].
(2) The **average-case execution time** (**ACET**) of a task is generated based on a Weibull distribution that fulfills the properties in [25, Table III, IV, and V].
(3) The task's WCET is determined by multiplying its ACET with its WCET factor, which is drawn equally distributed from an interval $[f_{min}, f_{max}]$ [25].

For the single ECU case, we generate 1,000 *automotive task sets* for each cumulative task set utilization of $U = 50\%, 60\%, 70\%, 80\%$, and 90%. Since the tasks' utilizations are determined by the WCET and the automotive specific semi-harmonic periods, we used a fully-polynomial approximation scheme to solve the subset-sum problem to select a subset of tasks within a candidate task set such that the cumulative utilization satisfies the above requirements. We initially generate $\mathcal{T}$, a set of 1,000 to 1,500 tasks, and then select a subset $\mathcal{T}'$ of tasks using the subset-sum approximation algorithm to reach the targeted utilization within 1 percentage point error bounds, i.e., $|(\sum_{\mathcal{T}'} U_i) - U| \leq 0.01$. On average, the generated task sets consist of 50 tasks.

Uniform task set generation [7]: For user-specified values $n \in \mathbb{N}$ and $0 < U^* \leq 1$, the UUniFast algorithm [7] draws utilizations $(U_1, U_2, \ldots, U_n)$ from $(0, 1]^n$ uniform at random under the constraint that $\sum_{i=1}^{n} U_i = U^*$. Due to the fact that the analyses are computationally tractable only for sufficiently small hyperperiods, we draw semi-harmonic periods based on the automotive benchmark. For each of the utilization values, i.e., $U^* = 50\%, 60\%, 70\%, 80\%$, and 90%, we generate 1,000 task sets with 50 tasks each. Each task's period is drawn from the interval $[1, 2, 000]$ according to a log-uniform distribution and rounded to the next smallest period in the set $\{1, 2, 5, 10, 20, 50, 100, 200, 500, 1{,}000\}$. Given the periods and utilizations, the WCET is set to $U_i \cdot T_i$.

To the best of our knowledge, there are no benchmarks published that detail and reason how to experimentally set up an asynchronous release of tasks, i.e., what a task's phase value should be. Furthermore, the analysis by Kloda et al. [22] is formulated only for cause-effect chains with synchronous tasks. Hence, we consider synchronous task sets (with $\phi_i = 0$) in this evaluation. All tasks are scheduled by preemptive **Rate Monotonic** (**RM**) scheduling.

7.2 Communication Tasks

In order to evaluate interconnected cause-effect chains, we assume a fixed-priority communication fabric. Specifically, we draw the period of each message log-uniform at random from the range 10 *ms* to 10,000 *ms* and truncate the result to the next smallest integer to model the communication frequency. Furthermore, we assume that the transmission time of a message, i.e., execution time on the communication fabric, is a constant. In our evaluations, we utilize the constant time from standard 2.0A CAN-Bus with 1 Mbps bandwidth, where transmitting 8 bytes of data (along with its 66 bits overhead due to its header and tail) takes $C_i = 130 \cdot 10^{-3}$ *ms*. Given the set of all messages and with random priority assignment, the WCRT of each communication task is calculated using time-demand analysis for non-preemptive tasks.

7.3 Cause-effect Chain Generation

Intra-ECU cause-effect chain generation: Given a generated task set as described in Section 7.1, the set of cause-effect chains is generated according to the description in Section IV-E in [25].

[3]The sum of the probabilities in [25, Table III, IV, and V] is only 85%. The remaining 15% is reserved for angle-synchronous tasks that we do not consider. Hence, all share values are divided by 0.85 in the generation process.

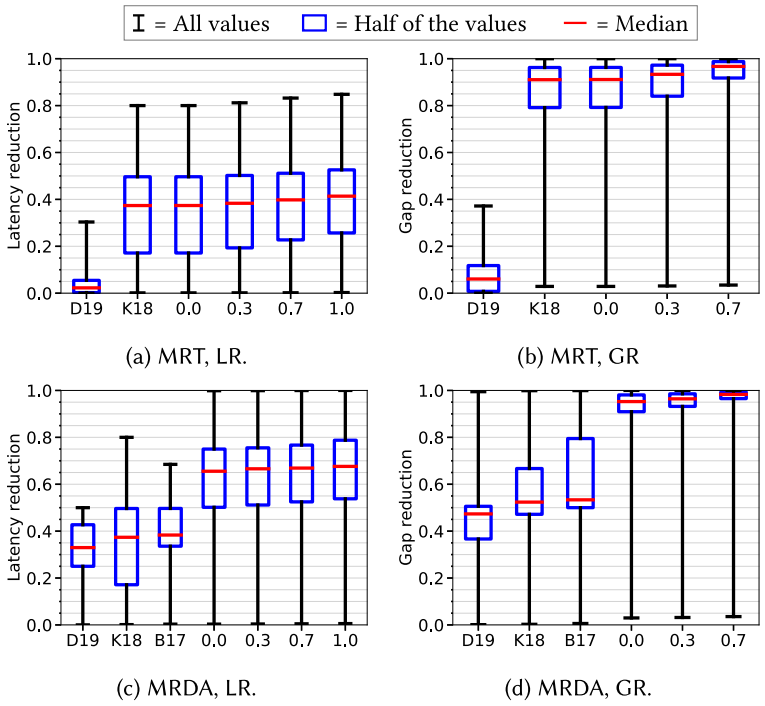


Fig. 9. **Intra-ECU** experiments for the **automotive benchmark**. Our method improves the state-of-the-art, especially for the MRDA.

Namely, a set of cause-effect chains containing 30 to 60 cause-effect chains is generated from each task set as depicted in the following steps:

(1) The number of involved activation patterns $P_j \in \{1, 2, 3\}$, i.e., the number of unique periods of tasks in a generated cause-effect chain, is drawn according to the distribution shown in Table VI in [25].
(2) P_j unique periods are drawn from the task set from a uniform distribution without replacement. More specifically, this step yields a set $\mathbf{T}_j$ of P_j distinct periods.
(3) For each period in $\mathbf{T}_j$ we draw 2 to 5 tasks at random (without replacement) according to the distribution in Table VII in [25] from the tasks in the task set with the respective period.

The resulting cause-effect chains consist of 2 to 15 tasks and no task occurs multiple times in the same cause-effect chain.

Inter-ECU cause-effect chain generation: We generate 10,000 interconnected cause-effect chains by selecting 5 cause-effect chains of different task sets with the same utilization under a uniform distribution. For each selection, we create 20 communication tasks as described in Section 7.2. Among them, 4 are chosen randomly to connect the 5 communication tasks.

7.4 Evaluation Results

In Figures 9 and 10, we show the evaluation results for the Intra-ECU case with automotive and uniform task generation, respectively. In Figures 11 and 12, we show the evaluation results for the Inter-ECU case with automotive and uniform task generation, respectively. Since our definition of *maximum reduced data age* from Section 6 coincides with the definition of data age from [4, 11,

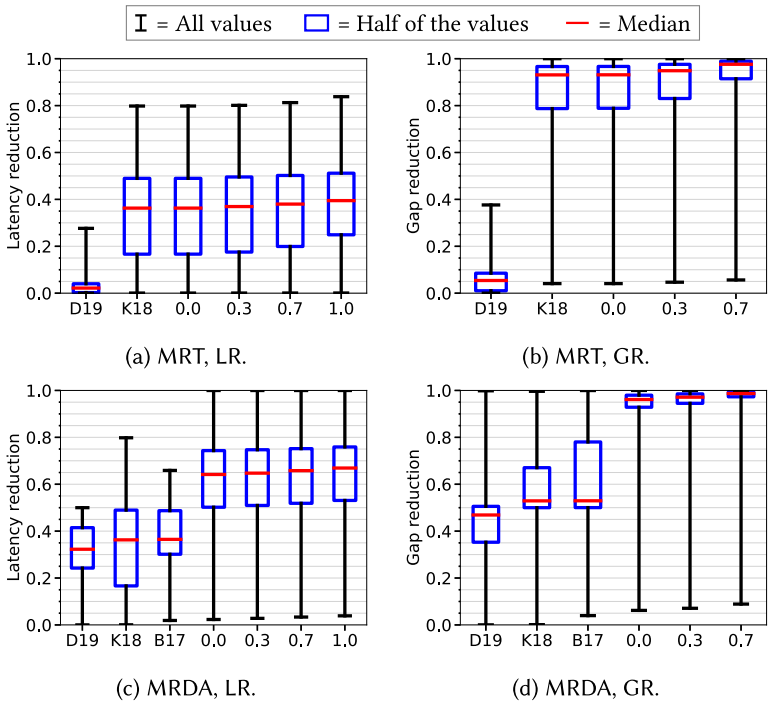


Fig. 10. **Intra-ECU** experiments for the **uniform benchmark**. Again, our method improves the state-of-the-art, especially for the MRDA.

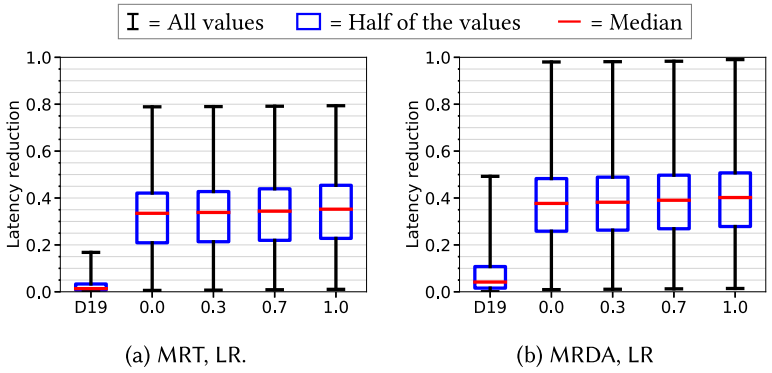
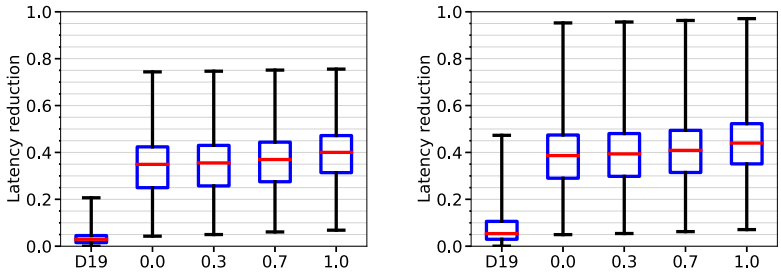


Fig. 11. **Inter-ECU** experiments for the **automotive benchmark**. Our method improves the state-of-the-art for all setups.

22], we compare our analysis using maximum reduced data age instead. The boxplots display the evaluation results of the methods by Dürr et al. [11] **(D19)** and Kloda et al. [22] **(K18)**, as well as of our method **(0.0, 0.3, 0.7, 1.0)** where the BCET of the task is set to 0.0, 0.3, 0.7 or 1.0 of the WCET. Since **1.0** is the bound given by Corollary 29, the gap reduction is always equal to 1 and therefore not reported. For the MRDA, the method by Becker et al. [4][4] **(B17)** is displayed as well. The plots show the **latency reduction** (**LR**) from Equation (29) or the **gap reduction** (**GR**) from

[4]We apply the method where the WCRT is known (see [4, Table 1]).



(a) MRT, LR. (b) MRDA, LR.

Fig. 12. **Inter-ECU** experiments for the **uniform benchmark**. Again, our method improves the state-of-the-art for all setups.

Equation (30). The method by Schlatow et al. [29] is omitted since it is dominated by Davare's method for non-harmonic task systems.

For the Intra-ECU case, our analysis performs similar to K18 for MRT ((a) and (b) in Figures 9 and 10) and outperforms the state-of-the-art for MRDA ((c) and (d) in Figures 9 and 10). We observe that under both benchmarks the LR and GR of our method increases when the BCET is closer to the WCET. The GR ((b) and (d) in Figures 9 and 10) is in median over 90 percent for our methods and for K18, whereas D19 and B17 have a median GR of less than 55 percent in all scenarios.

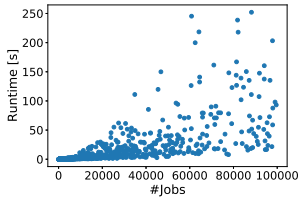
For the Inter-ECU case (Figures 11 and 12), our method outperforms the state-of-the-art significantly: Whereas our method shows a median LR of more than 30 percent in all cases, D19 has a median LR of around 5 percent or less.

7.5 Runtime Evaluation

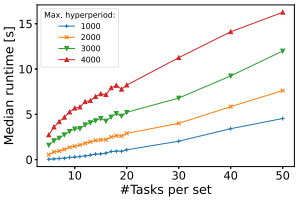
In the following, we study the control parameters that regulate the runtime of our analysis. More specifically, we show that (1) the runtime of our single ECU algorithm is dependent on the number of jobs to be scheduled in the simulation, and (2) the runtime can be controlled by bounding the hyperperiod of the task sets under analysis. For the measurements, we use a machine equipped with 2x AMD EPYC 7742 running Linux, i.e., in total 256 threads with 2,25 GHz and 256 GB RAM. Each measurement runs on one independent thread and covers the time for simulation of the best-case and worst-case schedule and for deriving the single ECU MRT, MDA, and maximum reduced data age. Both experiments rely on uniform task generation [7] with synchronous tasks. The total utilization is pulled uniformly from $[50, 90]$ [%] and task periods are pulled log-uniformly from the integers in $[1, 20]$. As a result, the hyperperiod of the task set is between 1 and $\text{lcm}(1, 2, \ldots, 20) = 232{,}792{,}560$. From each task set, we choose 5 tasks at random without replacement and combine them into a cause-effect chain.

For (1), we generate 1,000 task sets with 5 to 20 tasks each. The results are depicted in Figure 13(a) for task sets where the number of scheduled jobs is below 100,000. The number of scheduled jobs is upper bounded by $\#jobs \leq \sum_{i=1}^{\#tasks} \frac{2 \cdot hyp}{T_i} \leq \frac{2 \cdot hyp}{T_1} \cdot \#tasks$ as explained in Section 5.1. For a fixed number of tasks and a fixed range of periods, we can control the number of scheduled jobs by constraining the hyperperiod. For example, when the hyperperiod is 1,000 and the number of tasks is 20, then there are at most 40,000 scheduled jobs. The exact number of jobs may be lower since big hyperperiods require bigger periods T_i.

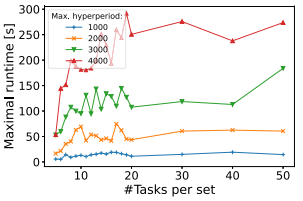
In (2), for a given number of tasks per set, we create 1,000 task sets with hyperperiod in the range from 0 to 1,000, 1,000 to 2,000, 2,000 to 3,000, and 3,000 to 4,000, each. The median and



(a) Dependency between number of jobs in the schedule and runtime of our analysis.



(b) Median runtime measurements of our analysis for different hyperperiod bounds.



(c) Maximal runtime measurements of our analysis for different hyperperiod bounds.

Fig. 13. Runtime evaluation.

maximal runtimes sorted by hyperperiod bounds are depicted in Figure 13(b) and (c), respectively. We observe that with a low hyperperiod, the maximum runtime can be controlled.

8 CONCLUSION

In this article, we analyze the *MRT* and *MDA*. We provide a precise definition in terms of augmented job chains and present a local analysis which performs close to the exact results. Moreover, we make this local analysis available to the interconnected ECU case by bounding the communication time between ECUs.

We plan to further explore priority assignments such that all cause-effect chains in a system meet their requirements. Moreover, we look for more efficient algorithms potentially obtained by partitioning cause-effect chains not only at the ECU-communication but also on one ECU.

APPENDIX

A BOUNDS FOR START AND FINISH

In the following we show that the latest starting time and finishing time of any job is achieved when all jobs execute their WCET, and the earliest starting time and finishing time is achieved when all jobs execute their BCET. In particular, we prove Propositions 13 and 14.

Proposition 13. *Let jc_{max} be the job collection of a set $\mathbb{T}$ of periodic tasks where all jobs execute according to their WCET. Consider the mth job $J = \tau(m, jc_{max})$ of a task $\tau \in \mathbb{T}$ in the job collection jc_{max}. Then, for the preemptive fixed-priority schedule $\mathcal{S}(jc_{max})$, the starting time and finishing time of J in $\mathcal{S}(jc_{max})$ are **upper bounds** on the starting time and finishing time of the mth job of τ for* any *job collection, respectively. That is, for any other job collection $jc \in JC(\mathbb{T})$, we have $s_J^{\mathcal{S}(jc_{max})} \geq s_{\tau(m,jc)}^{\mathcal{S}(jc)}$ and $f_J^{\mathcal{S}(jc_{max})} \geq f_{\tau(m,jc)}^{\mathcal{S}(jc)}$.*

Proposition 14. *Let jc_{min} be the job collection of a set $\mathbb{T}$ of periodic tasks where all jobs execute according to their BCET. Consider the mth job $J = \tau(m, jc_{min})$ of a task $\tau \in \mathbb{T}$ in the job collection jc_{min}. Then, for the preemptive fixed-priority schedule $\mathcal{S}(jc_{min})$, the starting time and finishing time of J in $\mathcal{S}(jc_{min})$ are **lower bounds** on the starting time and finishing time of the mth job of τ for* any *job collection, respectively. That is, for any other job collection $jc \in JC(\mathbb{T})$, we have $s_J^{\mathcal{S}(jc_{min})} \leq s_{\tau(m,jc)}^{\mathcal{S}(jc)}$ and $f_J^{\mathcal{S}(jc_{min})} \leq f_{\tau(m,jc)}^{\mathcal{S}(jc)}$.*

Proof of Propositions 13 and 14. Let jc be any job collection for $\mathbb{T}$. It is sufficient to show that for any job $J \in jc$ the following two properties hold:

Property P1. Whenever the job execution of any other job J' is increased (i.e., J' is replaced by a job with more execution time), then the starting time of J does not decrease.

Property P2. Whenever the job execution of any other job J' is increased, then the finishing time of J does not decrease. Or equivalently, whenever the job execution of any other job J' is *reduced*, then the finishing time of J does not increase.

The following proof is based on contradiction. In particular, we assume that P1 or P2 does *not* hold for all jobs in jc. We denote by $\tilde{J}$ the job in jc with the earliest finishing time such that P1 or P2 does not hold. In particular, for all jobs finished before $\tilde{J}$ in $\mathcal{S}(jc)$ both properties P1 and P2 hold. Let $\tilde{\tau}$ denote the task of $\tilde{J}$.

Without any impact on the execution behavior of $\tilde{J}$ we remove all jobs of tasks with lower priority than $\tilde{\tau}$. Moreover, we remove all jobs from $\tilde{\tau}$ that are released later than $\tilde{J}$.

We obtain a contradiction in two steps: First, we show that P1 holds for $\tilde{J}$, meaning that P2 does not hold for $\tilde{J}$. In the second step, we lead statement *"P2 does not hold for $\tilde{J}$"* to a contradiction.

Step 1: Proof that P1 holds for $\tilde{J}$. Let $[g_1, h_1)$ be the largest interval, such that

- $r(\tilde{J}) \in [g_1, h_1]$, and
- at all times during $[g_1, h_1)$ either jobs with higher priority than $\tilde{\tau}$ or jobs of $\tilde{\tau}$ released before $\tilde{J}$ are executed.

For this interval the property $s_{\tilde{J}}^{\mathcal{S}(jc)} = h_1$ holds. Let $\mathbb{J}_1$ be the set of jobs that are executed during $[g_1, h_1)$. Then during $[g_1, h_1)$ there is always pending workload from jobs of $\mathbb{J}_1$, i.e., for all $t \in [g_1, h_1)$ there exists some $J \in \mathbb{J}_1$ such that $t \in [r_J^{\mathcal{S}(jc)}, f_J^{\mathcal{S}(jc)})$. Moreover, for the jobs in $\mathbb{J}_1$ Properties P1 and P2 hold since those jobs finish before $\tilde{J}$ does.

When increasing the execution time of any job, then f_J does not decrease for all $J \in \mathbb{J}_1$ by Property P2. As a result, during the interval $[g_1, h_1)$ there is still pending workload from jobs of $\mathbb{J}_1$ at all times, i.e., the ECU is blocked and cannot execute $\tilde{J}$. Hence, $s_{\tilde{J}}^{\mathcal{S}(jc)}$ is still $\geq h_1$, i.e. P1 holds for $\tilde{J}$. Therefore, P2 does not hold for $\tilde{J}$ by the initial assumption.

Step 2: Contradiction of the statement *"P2 does not hold for $\tilde{J}$"*. Let $[g_2, h_2)$ be the largest interval, such that

(a) $\tilde{J}$ is executed at some time during the interval, and
(b) at all times in the interval (g_2, h_2) there is pending workload, i.e., workload that was released before but is not already finished, from $\tilde{J}$, earlier jobs of $\tilde{\tau}$ or jobs with higher priority than $\tilde{\tau}$.

Let $\mathbb{J}_2$ be the jobs that are executed during the interval $[g_2, h_2)$. By definition, $\tilde{J} \in \mathbb{J}_2$. In particular, (b) ensures that

$$g_2 + \sum_{\{J \,|\, J \in \mathbb{J}_2 \text{ and } r_J \in [g_2, e)\}} c_J > e \tag{31}$$

holds for all $e \in [g_2, h_2)$, where c_J is the actual execution time of job J, and

$$g_2 + \sum_{\{J \,|\, J \in \mathbb{J}_2 \text{ and } r_J \in [g_2, h_2)\}} c_J = h_2. \tag{32}$$

Moreover, we observe the typical busy interval properties $g_2 = \min_{J \in \mathbb{J}_2} r_J$, i.e., the busy interval starts with a job release, and

$$\min_{J \in \mathbb{I}} r_J + \sum_{J \in \mathbb{I}} c_J \leq h_2, \tag{33}$$

for all subsets $\mathbb{I} \subseteq \mathbb{J}_2$, i.e., all workload can be finished until the end of the busy interval.

We have $f_{\tilde{J}} = h_2$ as the following consideration shows:

— $\tilde{J}$ is executed during $[g_2, h_2)$ by (a). Therefore, $\tilde{J}$ has pending workload during $[g_2, h_2)$, i.e., $(r_{\tilde{J}}, f_{\tilde{J}}) \cap (g_2, h_2) \neq \emptyset$. Since $[g_2, h_2)$ is the *largest* interval with pending workload and there is pending workload during $(r_{\tilde{J}}, f_{\tilde{J}})$, we conclude $(r_{\tilde{J}}, f_{\tilde{J}}) \subseteq (g_2, h_2)$ and therefore $f_{\tilde{J}} \leq h_2$.
— Assume $f_{\tilde{J}} < h_2$ *for contradiction.* Let $\mathbb{I} \subseteq \mathbb{J}$ be the set of the jobs that are executed during the interval $[f_{\tilde{J}}, h_2) \neq \emptyset$. By definition, $\tilde{J} \notin \mathbb{I}$ holds. All jobs of $\mathbb{I}$ have higher priority than $\tilde{J}$ by b). If a job of $\mathbb{I}$ would be released before $f_{\tilde{J}}$ then it would finish before $f_{\tilde{J}}$ due to its priority. Therefore, all jobs of $\mathbb{I}$ are released at or after $f_{\tilde{J}}$. Hence, there is no pending workload at time $f_{\tilde{J}}$ which contradicts (b).

Since Property P2 does not hold for $\tilde{J}$, there is a scenario where the execution time of any job is reduced but the finishing time of $\tilde{J}$ is increased. We denote the job collection with that modified execution time by jc'. Let $[g_2', h_2')$ be the busy interval of $\tilde{J}$, i.e., the interval that fulfills (a) and (b), in the new schedule $\mathcal{S}(jc')$. Moreover, let $\mathbb{J}_2'$ be the jobs in the new schedule $\mathcal{S}(jc')$ that are executed during $[g_2', h_2')$.

Analogous to the discussion that $h_2 = f_{\tilde{J}}$ in the original schedule $\mathcal{S}(jc)$, $h_2' = f_{\tilde{J}}$ holds in the new schedule $\mathcal{S}(jc')$. Hence, by the assumption that Property P2 does not hold, we have $h_2' > h_2$. Moreover, Equations (31), (32), and (33) hold for the new scenario as well.

All jobs that are finished before g_2 in the original schedule $\mathcal{S}(jc)$ fulfill Property P2. Therefore, they are also finished before g_2 in the new schedule $\mathcal{S}(jc')$. Consequently, in $\mathcal{S}(jc')$ there is no pending workload at time g_2, and $g_2' \geq g_2$ holds. In the following we show that both different cases $g_2' = g_2$ and $g_2' > g_2$ lead to a contradiction.

— Case $g_2' = g_2$: In the new schedule $\mathcal{S}(jc')$ we have $g_2' + \sum_{J \in \mathbb{J}_2' \subset jc'} c_J > h_2$ by Equation (31) when choosing $e = h_2 \in [g_2', h_2')$. However, in the original schedule $\mathcal{S}(jc)$ we have $g_2 + \sum_{J \in \mathbb{J}_2 \subset jc} c_J = h_2$ by Equation (32). Since the execution time of jobs is only decreased, we have $g_2' + \sum_{J \in \mathbb{J}_2' \subset jc'} c_J \leq g_2 + \sum_{J \in \mathbb{J}_2 \subset jc} c_J$, which leads to a contradiction that $h_2 < h_2$ as follows:

$$h_2 < g_2' + \sum_{J \in \mathbb{J}_2' \subset jc'} c_J \leq g_2 + \sum_{J \in \mathbb{J}_2 \subset jc} c_J = h_2.$$

— Case $g_2' > g_2$: Let $\mathbb{I} \subseteq \mathbb{J}_2$ and $\mathbb{I}' \subseteq \mathbb{J}_2'$ be the jobs that are released during $[g_2', h_2)$ in the original schedule $\mathcal{S}(jc)$ and in the new schedule $\mathcal{S}(jc')$, respectively. As jc' and jc are job collections of a periodic task set $\mathbb{T}$, each job in jc' is represented by a corresponding job in jc with the same release time but may have smaller execution time. Therefore, each job in $\mathbb{I}$ has also a corresponding job in $\mathbb{I}'$ that has the same release time but may have smaller execution time. Hence, we have $\sum_{J \in \mathbb{I}} c_J \geq \sum_{J \in \mathbb{I}'} c_J$. By Equation (33) in the original schedule we obtain $g_2' + \sum_{J \in \mathbb{I}} c_J \leq h_2$. By Equation (31) in the new schedule we obtain $g_2' + \sum_{J \in \mathbb{I}'} c_J > h_2$. These two conditions lead to the fact that $\sum_{J \in \mathbb{I}} c_J < \sum_{J \in \mathbb{I}'} c_J$ which contradicts $\sum_{J \in \mathbb{I}} c_J \geq \sum_{J \in \mathbb{I}'} c_J$.

Since we have lead both cases $g_2' = g_2$ and $g_2' > g_2$ to a contradiction, this concludes Step 2. □

REFERENCES

[1] AUTOSAR. 2022. Specification of Timing Extensions, Release R22-11. (November 2022). https://www.autosar.org/fileadmin/standards/R22-11/CP/AUTOSAR_TPS_TimingExtensions.pdf. Access Date 2 May 2023.

[2] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. 2016. Mechaniser-a timing analysis and synthesis tool for multi-rate effect chains with job-level dependencies. In *Proceedings of the Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems.*

[3] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. 2016. Synthesizing job-level dependencies for automotive multi-rate effect chains. In *Proceedings of the International Conference on Embedded and Real-Time Computing Systems and Applications*. 159–169. DOI: https://doi.org/10.1109/RTCSA.2016.41

[4] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. 2017. End-to-end timing analysis of cause-effect chains in automotive embedded systems. *Journal of Systems Architecture - Embedded Systems Design* 80 (2017), 104–113. DOI: https://doi.org/10.1016/j.sysarc.2017.09.004

[5] Matthias Becker, Saad Mubeen, Dakshina Dasari, Moris Behnam, and Thomas Nolte. 2017. A generic framework facilitating early analysis of data propagation delays in multi-rate systems. In *Proceedings of the International Conference on Embedded and Real-Time Computing Systems and Applications*. 1–11. DOI: https://doi.org/10.1109/RTCSA.2017.8046323

[6] Albert Benveniste, Paul Caspi, Paul Le Guernic, Hervé Marchand, Jean-Pierre Talpin, and Stavros Tripakis. 2002. A protocol for loosely time-triggered architectures. In *Proceedings of the International Conference on Embedded Software*. 252–265. DOI: https://doi.org/10.1007/3-540-45828-X_19

[7] Enrico Bini and Giorgio C. Buttazzo. 2005. Measuring the performance of schedulability tests. *Real-Time Systems* 30, 1-2 (2005), 129–154. DOI: https://doi.org/10.1007/s11241-005-0507-9

[8] Bosch. 1991. Controller Area Network specification 2.0. (1991). http://esd.cs.ucr.edu/webres/can20.pdf. Access Date 2 May 2023.

[9] Hyunjong Choi, Mohsen Karimi, and Hyoseung Kim. 2020. Chain-based fixed-priority scheduling of loosely-dependent tasks. In *Proceedings of the International Conference on Computer Design*. 631–639. DOI: https://doi.org/10.1109/ICCD50377.2020.00109

[10] Abhijit Davare, Qi Zhu, Marco Di Natale, Claudio Pinello, Sri Kanajan, and Alberto L. Sangiovanni-Vincentelli. 2007. Period optimization for hard real-time distributed automotive systems. In *Proceedings of the Design Automation Conference*. 278–283. DOI: https://doi.org/10.1145/1278480.1278553

[11] Marco Dürr, Georg von der Brüggen, Kuan-Hsun Chen, and Jian-Jia Chen. 2019. End-to-end timing analysis of sporadic cause-effect chains in distributed systems. *ACM Transactions on Embedded Computing Systems (Special Issue for CASES)* 18, 5s (2019), 58:1–58:24. DOI: https://doi.org/10.1145/3358181

[12] Rolf Ernst, Leonie Ahrendts, and Kai Bjorn Gemlau. 2018. System level LET: Mastering cause-effect chains in distributed systems. In *Proceedings of the IECON- 44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 4084–4089. DOI: https://doi.org/10.1109/IECON.2018.8591550

[13] Nico Feiertag, Kai Richter, Johan Nordlander, and Jan Jonsson. 2009. A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics. In *Proceedings of the Workshop on Compositional Theory and Technology for Real-Time Embedded Systems*.

[14] FlexRay Consortium. 2005. FlexRay Communications System-Protocol Specification. (2005). https://svn.ipd.kit.edu/nlrp/public/FlexRay/FlexRay%E2%84%A2%20Protocol%20Specification%20Version%203.0.1.pdf. Access Date 2 May 2023.

[15] Julien Forget, Frédéric Boniol, and Claire Pagetti. 2017. Verifying end-to-end real-time constraints on multi-periodic models. In *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation*. 1–8. DOI: https://doi.org/10.1109/ETFA.2017.8247612

[16] Alain Girault, Christophe Prevot, Sophie Quinton, Rafik Henia, and Nicolas Sordon. 2018. Improving and estimating the precision of bounds on the worst-case latency of task chains. *IEEE Transactions on CAD of Integrated Circuits and Systems, (Special Issue for EMSOFT)* 37, 11 (2018), 2578–2589. DOI: https://doi.org/10.1109/TCAD.2018.2861016

[17] Pourya Gohari, Mitra Nasri, and Jeroen Voeten. 2022. Data-age analysis for multi-rate task chains under timing uncertainty. In *Proceedings of the 30th International Conference on Real-Time Networks and Systems*. ACM, 24–35. DOI: https://doi.org/10.1145/3534879.3534893

[18] Mario Günzel, Kuan-Hsun Chen, Niklas Ueter, Georg von der Brüggen, Marco Dürr, and Jian-Jia Chen. 2021. Timing analysis of asynchronized distributed cause-effect chains. In *Proceedings of the IEEE 27th Real-Time and Embedded Technology and Applications Symposium*. 40–52. DOI: https://doi.org/10.1109/RTAS52030.2021.00012

[19] Arne Hamann, Dakshina Dasari, Simon Kramer, Michael Pressler, and Falk Wurst. 2017. Communication centric design in complex automotive embedded systems. In *Proceedings of the Euromicro Conference on Real-Time Systems*. 10:1–10:20.

[20] Arne Hamann, Dakshina Dasari, Falk Wurst, Ignacio Sañudo, Nicola Capodieci, Paolo Burgio, and Marko Bertogna. 2019. WATERS industrial challenge 2019. In *Proceedings of the Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems*.

[21] Christoph M. Kirsch and Ana Sokolova. 2012. The logical execution time paradigm. In *Proceedings of the Advances in Real-Time Systems*. Springer, 103–120. DOI: https://doi.org/10.1007/978-3-642-24349-3_5

[22] Tomasz Kloda, Antoine Bertout, and Yves Sorel. 2018. Latency analysis for data chains of real-time periodic tasks. In *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation*. 360–367. DOI: https://doi.org/10.1109/ETFA.2018.8502498

[23] Hermann Kopetz. 1997. *Real-time Systems - Design Principles for Distributed Embedded Applications*. The Kluwer international series in engineering and computer science, Vol. 395. Kluwer.

[24] Alix Munier Kordon and Ning Tang. 2020. Evaluation of the age latency of a real-time communicating system using the LET paradigm. In *Proceedings of the Euromicro Conference on Real-Time Systems*. 20:1–20:20. DOI: https://doi.org/10.4230/LIPIcs.ECRTS.2020.20

[25] Simon Kramer, Dirk Ziegenbein, and Arne Hamann. 2015. Real world automotive benchmark for free. In *Proceedings of the International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*.

[26] Joseph Y.-T. Leung and Jennifer Whitehead. 1982. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation* 2, 4 (1982), 237–250. DOI: https://doi.org/10.1016/0166-5316(82)90024-4

[27] A. C. Rajeev, Swarup Mohalik, Manoj G. Dixit, Devesh B. Chokshi, and S. Ramesh. 2010. Schedulability and end-to-end latency in distributed ECU networks: Formal modeling and precise estimation. In *Proceedings of the International Conference on Embedded Software*. 129–138. DOI: https://doi.org/10.1145/1879021.1879039

[28] Johannes Schlatow and Rolf Ernst. 2016. Response-time analysis for task chains in communicating threads. In *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium*. 245–254. DOI: https://doi.org/10.1109/RTAS.2016.7461359

[29] Johannes Schlatow, Mischa Möstl, Sebastian Tobuschat, Tasuku Ishigooka, and Rolf Ernst. 2018. Data-age analysis and optimisation for cause-effect chains in automotive control systems. In *Proceedings of the IEEE International Symposium on Industrial Embedded Systems*. 1–9. DOI: https://doi.org/10.1109/SIES.2018.8442077

[30] TU Dortmund LS12. 2022. End-To-End Timing Analysis. (2022). Retrieved 9 February 2023 from https://github.com/tu-dortmund-ls12-rt/end-to-end_inter.

Received 25 March 2022; revised 9 February 2023; accepted 12 February 2023