Timing Analysis of Asynchronized Distributed Cause-Effect Chains

Mario Günzel, Kuan-Hsun Chen, Niklas Ueter, Georg von der Brüggen, Marco Dürr and Jian-Jia Chen

TU Dortmund, Department of Computer Science, Dortmund, Germany
https://ls12-www.cs.tu-dortmund.de/

Citation: 10.1109/RTAS52030.2021.00012

Timing Analysis of Asynchronized Distributed Cause-Effect Chains

Mario Günzel, Kuan-Hsun Chen, Niklas Ueter, Georg von der Brüggen,
Marco Dürr and Jian-Jia Chen
Department of Computer Science
Technical University of Dortmund
Dortmund, Germany
{mario.guenzel, kuan-hsun.chen, niklas.ueter, georg.von-der-brueggen, marco.duerr, jian-jia.chen}@tu-dortmund.de

***Abstract*—Real-time systems require the formal guarantee of timing-constraints, not only for the individual tasks but also for the data-propagation paths. A cause-effect chain describes the data flow among multiple tasks, e.g., from sensors to actuators, independent from the priority order of the tasks. In this paper, 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 (duration of data processing) and the maximum data age (worst-case data freshness). On one local electronic control unit (ECU), we present how to compute the exact local (worst-case) end-to-end latencies when the execution time of the periodic tasks is fixed. We further extend our analysis to globally asynchronized systems by combining 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 for periodic task activations.**

***Index Terms*—Real-Time Systems; Embedded Control Systems; Cause-effect Chains; End-to-end Timing Analysis; Distributed Systems;**

I. Introduction

In industrial systems with real-time constrains, timeliness is required to ensure the correct functionality of software operations. Specifically, timing properties like end-to-end latencies are used to validate safety-critical tasks that have to perform a desired controlling behavior within a certain time interval, e.g., the interaction of electronic control units (ECUs) in a car.

A *cause-effect chain* describes a sequence of tasks that are necessary to complete a specific functionality, e.g., the first task reads the sensor value, the second task processes the sensor value, and the third task produces an output based on the sensor's reading. The time interval from a cause to an effect must be determined to validate the timing requirements of the procedure, a so-called *end-to-end timing analysis*.

Most approaches in the literature that validate timing requirements of cause-effect chains can in be classified into two categories: active approaches [8], [15], [22], 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], [5], [9], [10], [12], [14], [17], [21], [23], that discuss how the data should be properly 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 analyzing cause-effect chains, two types of end-to-end latencies have been primarily considered in the literature: The *maximum reaction time* denotes the length of the longest time interval starting from the occurrence of an external cause to the earliest time where this external cause is fully processed, i.e., the maximum *button to action delay*. The *maximum data age* 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. We note that the maximum reaction time includes the time between sampling and the previous sampling, since the external cause may occur right after the previous sampling. The maximum reaction time (data age, respectively) of a schedule can be determined by computing the maximum length of immediate forward (backward, respectively) *job chains* as defined in [10]. However, this definition of maximum reaction time and maximum data age is limited to a certain sporadic task model, where each task has minimum and maximum inter-arrival time. In this work we provide a definition without bounding it to any specific task model. Moreover, in the literature [9], [10], [17] there is the (implicit) assumption to measure maximum reaction time and data age only when all tasks are already in the system, i.e., when all tasks have released their first job. We refine this assumption which enables a *compositional property*: Instead of computing maximum reaction time (data age) of a cause effect chain E we can decompose E into smaller segments and analyze each of them separately.

We present an exact end-to-end analysis for single ECUs, in which predefined periodic tasks are scheduled under a fixed priority preemptive 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) [7] or FlexRay [13]. 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. In this paper such globally asynchronized locally synchronized (GALS) systems are examined. We note that although the notion of *ECUs* is adopted from automotive systems, our work can be abstracted to similar settings.

To analyze the maximum reaction time and maximum data age, the following results are provided in the literature:

- An upper bound for the maximum reaction time of cause-effect chains for periodic task sets on GALS systems by Davare et al. [9].
- Two upper bounds, one for maximum reaction time and one for maximum data age of cause-effect chains for sporadic task sets on GALS systems by Dürr et al. [10].
- An upper bound for the maximum reaction time of cause-effect chains for periodic task sets on globally synchronized systems provided by Kloda et al. [17].

Since Dürr et al. [10] show that the maximum data age is less than or equal to the maximum reaction time, the upper bounds provided by Davare et al. [9] and Kloda et al. [17] also hold for the maximum data age. However, the analysis by Kloda et al. [17] is formulated only for synchronous task releases, i.e., the first job of each task is released at time 0, and assumes that the worst-case response time of each task is known beforehand. Moreover, the paper from Schlatow et al. [23] focuses on the analysis of maximum data age of harmonic task systems. The analysis for non-harmonic cases is in fact more pessimistic than Davare's analysis, i.e., Eq. (36) in [23] plus the worst-case response times of the tasks in the chain is dominated by the analysis in [9].

It is worth noting that Dürr et al. [10] showed that their proposed end-to-end timing analyses under the sporadic task model dominate the upper bound proposed by Davare et al. [9] analytically, which is used in several known analyses [2]–[4]. In this work, we leverage on their analytical upper bounds as a backbone and improve it when the actual-case execution time of a periodic task is *fixed*.

Contributions: We examine maximum reaction time and maximum data age of cause effect chains for asynchronous periodic task sets on globally asynchronized locally synchronized (GALS) distributed systems. Our contributions are:

- In Section IV we provide precise definitions of *maximum reaction time* and *maximum data age*. 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 or sporadic tasks, and communication models, e.g., implicit communication or logical execution time (LET). It covers the single ECU as well as the interconnected ECU scenario.
- In Section V-B we provide a method to determine exact maximum reaction time and maximum data age in the single ECU scenario with periodic task sets under preemptive fixed priority scheduling if the execution time for each job is fixed. In particular we show that in this case it is sufficient to evaluate only the job chains released in a bounded time window.
- In Section V-C we extend the exact local analysis to the interconnected ECU scenario and present a method to bound the time for communication between ECUs in a globally asynchronized distributed system.
- We evaluate our proposed analysis for the single and interconnected ECU case in Section VII by comparing it with results from the literature, showing that our method outperforms state-of-the-art analyses for both maximum reaction time and maximum data age.

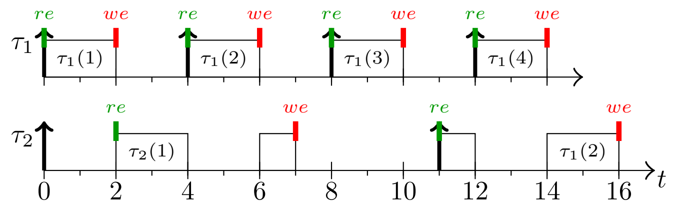


Figure 1: Read and write events under implicit communication.

II. System Model

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

A. Jobs and Tasks

For a general definition of maximum reaction time and maximum data age we rely on a very basic model of jobs and tasks. If the electronic control units (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., an ECU is equivalent to the classical 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 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. In this work 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 IV are valid for all kinds of task models, e.g., periodic or sporadic, the analysis in Section V is limited to *periodic tasks*. A periodic task is described by the tuple $\tau = (C_\tau, T_\tau, \phi_\tau) \in \mathbb{R}^3$, where $C_\tau \geq 0$ is the worst-case execution time (WCET) of the tasks, i.e., the longest runtime of a task without preemption or interrupt, $T_\tau > 0$ the period, and $\phi_\tau \geq 0$ the phase of the task. The first job is released at time ϕ_τ. Afterwards, 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 $c_{\tau(m)} \in [0, C_\tau]$ for all $m \in \mathbb{N}$. The utilization of a task τ is defined by $U_\tau := \frac{C_\tau}{T_\tau}$. 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 hyperperiod of a task set $\mathbb{T}$

is $H = \text{lcm}(\{T_\tau \mid \tau \in \mathbb{T}\})$, i.e., the least common multiple of all periods in the system. The existence of such hyperperiod is required for our analysis in Section V.

B. Communication Model

To ensure communication between different jobs, they receive (read) their input from a shared resource and hand over (write) their output to a shared resource. 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 $we_J^{\mathcal{S}}$. 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, i.e., $re_J^{\mathcal{S}} = s_J^{\mathcal{S}}$ and $we_J^{\mathcal{S}} = f_J^{\mathcal{S}}$, as depicted in Figure 1. The other one is based on the concept of *logical execution time (LET)* [16]. 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* $d_J = r_J + D_\tau$ and the read- and write-events of J are set to its release time and deadline, i.e., $re_J^{\mathcal{S}} = r_J$ and $we_J^{\mathcal{S}} = d_J$. Although LET is originally limited to single ECU, Ernst el al. [11] 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 $\mathcal{S}$. We assume that the following (not very restrictive) requirements are met:

- The read- and write-events are ordered in the sense that $re_{\tau(m)} < re_{\tau(m+1)}$, $we_{\tau(m)} < we_{\tau(m+1)}$ and $re_{\tau(m)} \leq we_{\tau(m)}$ for all $m \in \mathbb{N}$.
- The sets $\{re_{\tau(m)} \mid m \in \mathbb{N}\}$ and $\{we_{\tau(m)} \mid m \in \mathbb{N}\}$ 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 standard task models with periodic or sporadic tasks together with LET or implicit job communication.

Standard applications are composed of multiple ECUs. We model the communication infrastructure between different ECUs by additional *communication tasks* τ^c. Those are usual tasks in the sense of Section II-A where each job has the purpose of transferring data between ECUs. More specifically, jobs released by communications tasks read data from a shared resource of one ECU and write is to a shared resource of another ECU. Most communication tasks are modeled to be executed on additional communication ECUs, such that they do not impair job execution on the remaining ECUs.

C. Cause-Effect Chains

To serve a certain purpose, data has to be processed by different programs successively. A *cause-effect chain* $E = (\tau_1 \to \cdots \to \tau_k)$ 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 $E = (\tau_3 \to \tau_1)$. We note that if the tasks in the system are prioritized, e.g., for the use of some scheduling algorithm, the cause-effect chain does not necessarily follow this order.

We denote by $|E|$ the number of tasks in E, where $|E| \geq 1$. The function $E()$ returns for $m \in \{1, \dots, |E|\}$ the m-th task of the cause-effect chain E. For example, let $E = (\tau_4 \to \tau_5 \to \tau_1)$, 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*. In this work 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 a *sampling task* τ^{sample} to the system, where each job $\tau^{sample}(1), \tau^{sample}(2), \dots$ reads and writes data at a time where the sampling happens. Please note that the read- and write-events of the jobs of τ^{sample} need to fulfill the properties from the previous subsection and that the additional task should not affect the schedule $\mathcal{S}$, e.g., by assigning it to an additional ECU or by giving it a WCET of 0.

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 IV are valid for all kinds of cause-effect chains. The distinction is only necessary for the analysis in Section V.

D. Job Chains

The concept of job chains is essential to determine maximum reaction time and maximum data age. We adapt the definition from [10] to our model with read- and write-events. Let E and $\mathcal{S}$ be a cause-effect chain and a schedule for $\mathbb{T}$.

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

- The entry J_i is a job of $E(i)$ for all $i \in \{1, \dots, |E|\}$.
- 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 $i \in \{1, \dots, |E| - 1\}$.

Similar to [10], 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 $jc^{E,\mathcal{S}} = (J_1, \dots, J_{|E|})$ where for all $i \in \{1, 2, \dots, |E| - 1\}$ the read-event of the job J_{i+1} is the earliest after the write-event of the job J_i, i.e., $J_{i+1} = \arg\min_{J \in E(i+1), re_J \geq we_{J_i}} re_J$.

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

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

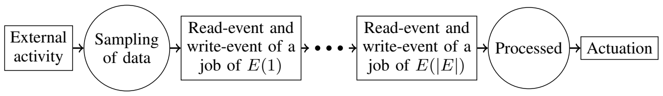

External activity
Sampling of data
Read-event and write-event of a job of $E(1)$
Read-event and write-event of a job of $E(|E|)$
Processed
Actuation

Figure 2: Chain of events to trace one data stream of E.

III. Problem Definition

In this paper we analyze *maximum reaction time* and *maximum data age* of distributed cause-effect chains E on globally asynchronized locally synchronized (GALS) systems. We assume that to each electronic control unit (ECU) the associated task set is known beforehand and that further communication tasks between the ECUs are given.

- **Input:** Some (interconnected) cause-effect chain E.
- **Output:** An upper bound on the maximum reaction time and the maximum data age of E that is obtained by introducing fixed execution time on each (non-communication) ECU.

To solve this problem, we 1) provide an exact bound for the local case in Section V-B and 2) extend the analysis to several ECUs in Section V-C. Please note that although we need to introduce fixed execution time into the system to perform the analysis, as a result the estimation becomes tighter as discussed in Section V-A and demonstrated in Section VII.

IV. Maximum Reaction Time and Data Age

This paper 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. End-to-end timing is important to guarantee the correct functionality of safety critical tasks within a given time frame. For control engineering the *maximum reaction time* (How long does it take until an external cause is processed?) and the *maximum data age* (How old is the data used in an actuation?) of a cause-effect chain are of special interest.

A. Augmented Job Chains

Let E and S be a cause-effect chain and 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 summarized in Figure 2: The (change of) data is provoked by some *external activity* and is fed into the system by *sampling*. By our assumption in Section II-C, the sampling coincides with the read-event of a job of the first task $E(1)$. After the first job in the sequence of events finishes execution, it writes the data to a shared resource. Afterwards, the second job reads the data from that shared resource, processes it, and writes it again to a shared resource for the next task. 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.

Maximum data age and reaction time are defined in [10] by using backward and forward job chains. In fact, job chains describe only the data stream from sampling until the data is processed. In this paper 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 $\mathcal{S}$ is a sequence $c^{E,\mathcal{S}} = (z, J_1, \dots, J_{|E|}, z')$, where $(J_1, \dots, J_{|E|})$ 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.

We denote by $c^{E,\mathcal{S}}(k)$ the k-th entry of the augmented job chain $c^{E,\mathcal{S}}$, i.e., $c^{E,\mathcal{S}}(1) = z$, $c^{E,\mathcal{S}}(|E|+2) = z'$ 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

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

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

Maximum reaction time measures timing from external activity to the instant where data is completely processed by the system. We omit time between 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 measure maximum reaction time, in the following way:

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

- The external activity happens directly after the m-th 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 $(J_1, \dots, J_E)$ 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 in the next subsection directly yields the definition of reaction time.

On the other hand, maximum data age measures time from sampling of data to an actuation based on that sampling. The actuation based on the data processed at a certain time happens directly before the next processed-event in the worst case.

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

- The actuation happens directly before the m-th processed-event, i.e., $z' = we_{E(|E|)(m)}$.
- The processed-event happens at the previous write-event of $E(|E|)$, i.e., $J_{|E|} = E(|E|)(m-1)$.

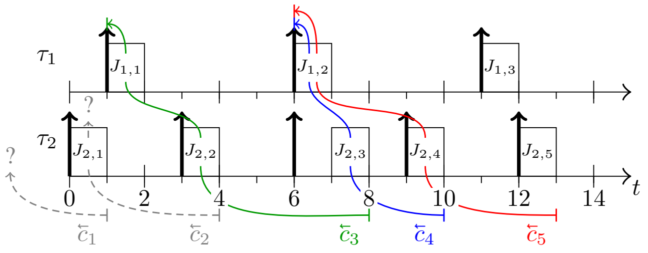


Figure 3: An example of backward augmented job chains under implicit communication. The cause-effect chain under analysis is $E = (\tau_1 \to \tau_2)$.

- 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 not for all $m \in \mathbb{N}$ there is an immediate backward augmented job chain. We call such chains *incomplete backward augmented job chains*. The length of incomplete augmented job chains is set to 0.

Example 7 (Backward augmented job chain determination). Figure 3 shows a single ECU schedule with two periodic tasks $\tau_1 = (C_{\tau_1} = 1, T_{\tau_1} = 5, \phi_{\tau_1} = 1)$ and $\tau_2 = (C_{\tau_2} = 1, T_{\tau_2} = 3, \phi_{\tau_2} = 0)$. 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 $E = (\tau_1 \to \tau_2)$ starts with 5-th read 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 also incomplete since there is no immediate backward job chain with 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 representation.

B. Definition of Maximum Reaction Time and Data Age

Let E and $\mathcal{S}$ be a cause-effect chain and a schedule with task set $\mathbb{T}$. With respect to Figure 2,

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

We note that our definition may differ from other definitions in the literature, e.g., in [10] data age only covers the time until the data is processed. Our definition of maximum data age is necessary to obtain a compositional property in Section IV-C,

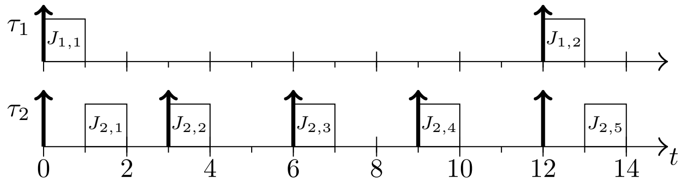


Figure 4: Under implicit communication, the second task has slightly shifted read-event.

i.e., that the maximum data age of a cause-effect chain E is the sum of the maximum data age of the segments of E. However, we discuss in Section VI how our analysis can be applied to the definition provided in [10].

Similar to [10], we could define maximum reaction time (data age) by taking the supremum of the length of *all* immediate forward (backward) augmented job chains. However, due to shifting of first read-event, e.g., induced by phase shifting, the reaction time might become arbitrarily large:

Example 8. 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 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.,

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

This property is similar to the (implicit) assumption in [10] to measure maximum reaction time and data age only when all tasks are already in the system. However, this way a slight shift of only one read-event might exclude many augmented job chains from consideration:

Example 9. For the schedule from Figure 4, if the jobs adhere implicit communication, the read-event of the first job of τ_2 is slightly shifted. With the approach from Eq. (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 $\vec{c}_1, \vec{c}_2, \vec{c}_3, \vec{c}_4$, and $\vec{c}_5$ would be reasonable to include.

To tackle this problem, 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 define such augmented job chains as *valid*:

Definition 10 (Valid)**.** Let $c^{E,\mathcal{S}} = (z, J_1, \ldots, J_{|E|}, z')$ 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 maximum reaction time and maximum data age.

Definition 11 (Maximum reaction time and data age)**.** For a cause-effect chain E with schedule $\mathcal{S}$ we define the schedule specific *maximum reaction time* and *maximum data age* by

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

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

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

Please note that this definition holds for all different 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 maximum reaction time 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 maximum reaction time* and *maximum data age*

$$\text{Reaction}(E) := \sup_{\mathcal{S}} \text{Reaction}(E,\mathcal{S}) \quad (5)$$

$$\text{DataAge}(E) := \sup_{\mathcal{S}} \text{DataAge}(E,\mathcal{S}) \quad (6)$$

by the supremum over all possible schedules $\mathcal{S}$.

In their Theorem 6.2, Dürr et al. [10] prove that the data age is bounded by the reaction time for their system model. We note that even for this generalized definition,

$$\text{DataAge}(E,\mathcal{S}) \leq \text{Reaction}(E,\mathcal{S}) \quad (7)$$

holds: 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 p-th job of task $E(1)$, i.e., $\overleftarrow{c}_m^{E,\mathcal{S}}(1) = re_{E(1)(p)}$. Similar to the proof of Theorem 6.2 in [10] we show that

$$\ell(\overleftarrow{c}_m^{E,\mathcal{S}}) \leq \ell(\vec{c}_p^{E,\mathcal{S}}), \quad (8)$$

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 $\text{DataAge}(E) \leq \text{Reaction}(E)$ since Eq. (7) holds for all possible schedules $\mathcal{S}$.

C. Cutting of Augmented Job Chains

One essential ingredient to apply a local analysis to the interconnected case in Section V-C is to cut the cause-effect chain into smaller (local) parts. Our definition of maximum reaction time and data age enables the possibility to deduce upper bounds by determining maximum reaction time and data age on the smaller segments. We prove in this section that this compositional property holds, even without restricting to any task or communication model and without introducing fixed execution time.

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

$$\textit{Reaction}(E,\mathcal{S}) \leq \textit{Reaction}(E_1,\mathcal{S}) + \textit{Reaction}(E_2,\mathcal{S}) \quad (9)$$

$$\textit{DataAge}(E,\mathcal{S}) \leq \textit{DataAge}(E_1,\mathcal{S}) + \textit{DataAge}(E_2,\mathcal{S}) \quad (10)$$

for any schedule $\mathcal{S}$.

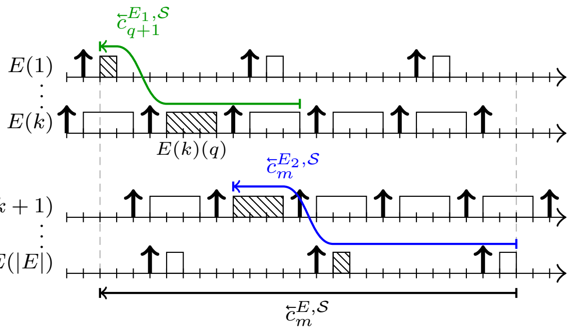

$\overleftarrow{c}_{q+1}^{E_1,\mathcal{S}}$
$E(1)$
$E(k)$
$E(k)(q)$
$\overleftarrow{c}_m^{E_2,\mathcal{S}}$
$E(k+1)$
$E(|E|)$
$\overleftarrow{c}_m^{E,\mathcal{S}}$

Figure 5: Cutting one immediate backward augmented job chain $\overleftarrow{c}_m^{E,\mathcal{S}}$ into two as in the proof of Theorem 12 (Cutting).

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. In Figure 5 the procedure is presented for immediate *backward* augmented job chains, assuming that 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 events for external event and actuation have to be determined properly.

Proof of Theorem 12 (Cutting): We first prove Eq. (10). By definition, $\text{DataAge}(E,\mathcal{S})$ is the supremum of the length of all valid immediate backward augmented job chains. Let $\overleftarrow{c}_m^{E,\mathcal{S}} = (re_{J_1}, J_1, \dots, J_{|E|}, z')$ with $m\in\mathbb{N}$ be some valid immediate backward augmented job chain. Let further $q\in\mathbb{N}$ such that J_k is the q-th write event of task $E(k)$, i.e., $E(k)(q) = J_k$. The scenario is depicted in Figure 5. By 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, $(re_{J_1}, J_1, \dots, J_k, \tilde{z}') = \overleftarrow{c}_{q+1}^{E_1,\mathcal{S}}$ and $(re_{J_{k+1}}, J_{k+1}, \dots, J_{|E|}, z') = \overleftarrow{c}_m^{E_2,\mathcal{S}}$ 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

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

Applying the supremum yields the result from Eq. (10).

Analogously we prove (9). By definition $\text{Reaction}(E,\mathcal{S})$ is the supremum of the length of all valid immediate forward augmented job chains. Let $\vec{c}_m^{E,\mathcal{S}} = (z, J_1, \dots, J_{|E|}, we_{J_{|E|}})$ with $m\in\mathbb{N}$ be some valid immediate forward augmented job chain. Let further $p \in \mathbb{N}$ 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.

With the definition of $\tilde{z}$ from above, we define two immediate forward augmented job chains $(z, J_1, \dots, J_k, we_{J_k}) = \vec{c}_m^{E_1,\mathcal{S}}$ and $(\tilde{z}, J_{k+1}, \dots, J_{|E|}, we_{J_{|E|}}) = \vec{c}_{p-1}^{E_2,\mathcal{S}}$. 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,

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

Applying the supremum yields the result from Eq. (9). ■

Since Eq. (9) and (10) hold for all schedules $\mathcal{S}$, the Cutting-Theorem does also hold for the overall maximum reaction time and overall maximum data age, i.e., $\text{Reaction}(E) \leq \text{Reaction}(E_1) + \text{Reaction}(E_2)$ and $\text{DataAge}(E) \leq \text{DataAge}(E_1) + \text{DataAge}(E_2)$. This compositional property can deal with clock-shifts by cutting at those positions where clock shifts occur.

V. Analysis of End-To-End Latencies

In this section we assume that the tasks on each ECU are scheduled according to preemptive fixed priority scheduling, i.e., they have a fixed priority-ordering and at each time the pending jobs of the task with the highest priority are executed. Our objective is to determine maximum reaction time and maximum data age of such systems.

Consider a schedule $\mathcal{S}$ of a task system $\mathbb{T}$ as above and a cause-effect chain E of $\mathbb{T}$. If the pattern of read- and write-events in $\mathcal{S}$ repeats after a certain amount of time, it suffices to analyze only a limited time window to compute maximum reaction time and maximum data age. Depending on the communication policy, there are different ways how to achieve a recurrent pattern of read- and write-events.

For **LET**, the read- and write-events repeat each hyperperiod after the maximal first read (which is at the maximal phase $\Phi := \max_\tau \phi_\tau$). 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 find 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 [18] present a way to compute maximum data age on single ECU systems efficiently for LET, using this procedure as a backbone.

If we assume **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. In this case we fix the execution time of each job to the worst-case execution time to obtain an recurrent schedule. This can be achieved by spinning early completed jobs until their worst-case execution time is reached. The remainder of this section shows that these properties suffice to make maximum reaction time and maximum data age computable. Although fixing the execution time of each job seems to be a bad idea on first sight, this actually improves the analysis as outlined in Section V-A. With similar argumentation as provided here, any communication policy which is only based on the start and finish of the jobs can be handled. Please note that the existence of a small hyperperiod is mandatory to compute data age and reaction time in acceptable time. In the following we further investigate how to compute maximum reaction time and data age for implicit communication with fixed execution time.

A. *Motivation for Fixed Execution Time*

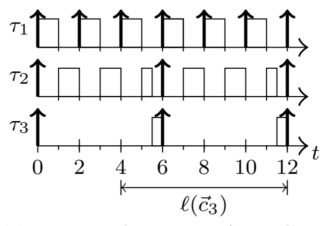
At first glance, fixing execution of jobs to their worst case might seem to be a bad idea, since our intuition says that this also increases end-to-end latencies of cause-effect chains for those tasks. However, this is not always the case. In fact, there are good reasons to forbid early completion of jobs as depicted below, and the schedulability of the task set is not affected. Therefore, provoking fixed execution time should be considered as a system design decision.

The first reason is that there are indeed cases where early completion increases end-to-end latencies. Consider for example the periodic task set $\mathbb{T} = \{\tau_1, \tau_2, \tau_3\}$ with tasks $\tau_1 = (C_{\tau_1} = 1, T_{\tau_1} = 2, \phi_{\tau_1} = 0)$, $\tau_2 = (C_{\tau_2} = 2.5, T_{\tau_2} = 6, \phi_{\tau_2} = 0)$ and $\tau_3 = (C_{\tau_3} = 0.5, T_{\tau_3} = 6, \phi_{\tau_3} = 0)$, and the cause-effect chain $E = (\tau_1 \to \tau_3)$. As depicted in Figure 6, early completion of τ_2 leads to a worse result. Both $\vec{c}_1$ and $\vec{c}_2$ in the left schedule have actuation at time 6. $\vec{c}_2$ is the longest immediate forward augmented job chain. In the right schedule the first read event of τ_3 is shifted to the left and therefore $\vec{c}_1$ has actuation at time 12.

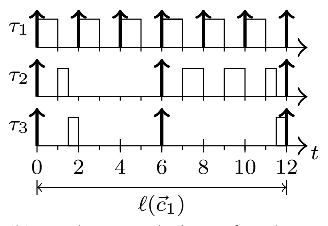
Secondly, by provoking fixed execution time for jobs in the following sections, we create a tight (even exact in the single ECU case) analysis and obtain superior *latency guarantees*, even if the average end-to-end latency might increase. For correct functionality of safety critical systems, e.g. in the automotive domain, timing guarantees of end-to-end latencies are much more important than the average behavior.

B. *Local Analysis*

In this section we assume that the cause-effect chain E under analysis is local, i.e., it contains only tasks on one (synchronized) ECU. Under the assumption that there is no early completion, there is a unique schedule for any periodic task set. We show that the schedule repeats after a certain amount of time. Hence, under implicit communication also the read- and write events repeat. The following considerations are



(a) Every task executes its WCET.



(b) Early completion of task τ_2.

Figure 6: Two schedules of jobs released periodically by 3 tasks. For $E = (\tau_1 \to \tau_3)$, early completion of the first job of τ_2 leads to a larger immediate forward augmented job chain.

based on the work of Leung and Whitehead [20]. Their proofs can not 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.

Let $\mathcal{S}$ be the fixed-priority schedule of the task set $\mathbb{T} = \{\tau_1, \dots, \tau_n\}$ on one ECU where the execution of all jobs is fixed to their worst-case. 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 $i < j$. Furthermore, let $\Phi := \max_{\tau \in \mathbb{T}} \phi_\tau$ be the maximal phase of tasks in $\mathbb{T}$. For a time instant t, we denote by $\mathcal{S}(t)$ the tuple $(s_{1,t}, \dots, s_{n,t})$ where each $s_{i,t}$ is the amount of time that jobs of τ_i were executed since their last release. Similar to the proof of [20, Lemma 3.3], we show the following.

Lemma 13. *Let Φ be the maximal phase of the task set $\mathbb{T}$ and $H = lcm(T_{\tau_1}, \dots, T_{\tau_n})$ the hyperperiod. Then for all $t \geq \Phi$ the relation $\mathcal{S}(t) \geq \mathcal{S}(t+H)$ (component-wise) holds.*

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

$$t_j \geq \phi_{\tau_j} \text{ with } s_{j,t_j} < s_{j,t_j+H}. \tag{11}$$

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

By assumption there is at least one task with the property from Eq. (11). 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., $s_{j',t'} < s_{j',t'+H} = C_{\tau_{j'}}$. 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 Eq. (11). ■

Furthermore, similar to [20, Lemma 3.4] we use the preceding lemma to show that the schedule repeats after $\Phi + 2H$, i.e., 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 II.

Lemma 14. *When $U_\mathbb{T} \leq 1$, then $\mathcal{S}(t) = \mathcal{S}(t+H)$ holds for all $t \geq \Phi + H$.*

Proof: We assume that there is some $t \geq \Phi + H$ with $\mathcal{S}(t) \neq \mathcal{S}(t+H)$. Then by Lemma 13 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 13 $\mathcal{S}(t') = (C_{\tau_1}, \dots, C_{\tau_n}) \leq \mathcal{S}(t'-H)$, 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, $\mathcal{S}(t) = \mathcal{S}(t+H)$.

For (b), since $s_{j,t} > s_{j,t+H}$ and $s_{i,t} \geq s_{i,t+H}$ for all i by Lemma 13, 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 $\sum_{i=1}^n C_{\tau_i} \frac{H}{T_{\tau_i}} > H$, which contradicts $\sum_{i=1}^n \frac{C_{\tau_i}}{T_{\tau_i}} \leq 1$. ■

Based on Lemma 14, the schedule repeats after $\Phi + 2H$. We conclude that the length of any immediate forward (backward) augmented job chain $\vec{c}_p^{E,\mathcal{S}}$ ($\overleftarrow{c}_m^{E,\mathcal{S}}$) with event for external activity at time $\vec{c}_p^{E,\mathcal{S}}(1) \geq \Phi + 2H$ ($\overleftarrow{c}_m^{E,\mathcal{S}}(1) \geq \Phi + 2H$) coincides with the length of the immediate forward (backward) augmented job chain whose event for external activity occurs one hyperperiod earlier. In particular, we have $\ell(\vec{c}_p^{E,\mathcal{S}}) = \ell(\vec{c}_{p'}^{E,\mathcal{S}})$ with $p' := p - \frac{H}{T_{E(1)}}$ and $\ell(\overleftarrow{c}_m^{E,\mathcal{S}}) = \ell(\overleftarrow{c}_{m'}^{E,\mathcal{S}})$ with $m' := m - \frac{H}{T_{E(|E|)}}$. Furthermore, $\vec{c}_{p'}^{E,\mathcal{S}}$ and $\overleftarrow{c}_{m'}^{E,\mathcal{S}}$ are valid as well since their event for external activity is at or after $\Phi + H$. We obtain that Reaction$(E, \mathcal{S})$ and DataAge$(E, \mathcal{S})$ can be expressed by a maximum of finitely many values:

$$\begin{aligned} \text{Reaction}(E,\mathcal{S}) &= \sup \left\{ \ell(\vec{c}_m^{E,\mathcal{S}}) \,\middle|\, m \in \mathbb{N}, \vec{c}_m^{E,\mathcal{S}} \text{ valid} \right\} \\ &= \max \left\{ \ell(\vec{c}_m^{E,\mathcal{S}}) \,\middle|\, \begin{matrix} m \in \mathbb{N}, \vec{c}_m^{E,\mathcal{S}} \text{ valid}, \\ \vec{c}_m^{E,\mathcal{S}}(1) < \Phi + 2H \end{matrix} \right\} \end{aligned} \tag{12}$$

$$\begin{aligned} \text{DataAge}(E,\mathcal{S}) &= \sup \left\{ \ell(\overleftarrow{c}_m^{E,\mathcal{S}}) \,\middle|\, m \in \mathbb{N}, \overleftarrow{c}_m^{E,\mathcal{S}} \text{ valid} \right\} \\ &= \max \left\{ \ell(\overleftarrow{c}_m^{E,\mathcal{S}}) \,\middle|\, \begin{matrix} m \in \mathbb{N}, \overleftarrow{c}_m^{E,\mathcal{S}} \text{ valid}, \\ \overleftarrow{c}_m^{E,\mathcal{S}}(1) < \Phi + 2H \end{matrix} \right\} \end{aligned} \tag{13}$$

We conclude that under the assumption that there is no early completion, reaction time and data age can be exactly computed by simulating the schedule.

Algorithm (Compute reaction time):

- Simulate all immediate forward job chains whose event for external activity occurs before $\Phi + 2H$.
- Compute reaction time with the formula in (12).

Algorithm (Compute data age):

- Simulate all immediate backward job chains whose event for external activity occurs before $\Phi + 2H$.
- Compute data age with the formula in (13).

Complexity: In our analysis, there are two components, which play decisive roles in time complexity. At first we have to (a) create the schedule for the bounded time frame, and then (b) create and compare job chains based on the schedule. We examine the time complexity for a cause-effect chain E on an ECU with task set $\mathbb{T} = \{\tau_1, \dots, \tau_n\}$ with n tasks.

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

For each augmented job chain we have to determine $|E|$ jobs and the events for external activity and actuation. There is a cost of query[1] Q depending on the data structure for finding the next job for the job chain. To compute the maximum reaction time, we have to simulate and compare up to $\frac{\Phi+2H}{T_{E(1)}}$ immediate forward job chains, and for the maximum data age up to $\frac{\Phi+2H}{T_{E(|E|)}}$ immediate backward job chains are under analysis. Hence, the time complexity for component (b) can be described by $\mathcal{O}\left(|E| \cdot Q \cdot \frac{\Phi+2H}{T_{E(1)}}\right)$ for reaction time and $\mathcal{O}\left(|E| \cdot Q \cdot \frac{\Phi+2H}{T_{E(|E|)}}\right)$ for data age.

We note that the time complexity for the method by Kloda et al. [17] 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. [10] and by Davare [9] have complexity $\mathcal{O}(|E|)$. Since in [9], [10], [17] they assume that the worst-case response times are known, i.e., computed in advance, the time complexity for this computation should also be taken into account.

C. 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 periodic tasks, then the analysis from the preceding subsection can be utilized. More specifically, we can either use LET or force fixed execution time for all tasks to reduce the computation of maximum reaction time and data age to a computation of finitely many augmented job chains.

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, i.e., these tasks may behave in a non-preemptive or even non-periodic manner. This hinders 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 worst-case response time R_{τ^c} and maximum inter-arrival time $T_{\tau^c}^{max}$, i.e., maximum time between two recurrent job releases, of each communication task τ^c.

[1]The data structure can be designed such that $Q = \mathcal{O}(1)$. 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 starting 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 $\left\lceil \frac{t-\phi_\tau - T_\tau}{T_\tau} \right\rceil$ and $\left\lfloor \frac{t-\phi_\tau}{T_\tau} \right\rfloor$ are candidates to be checked, i.e., $\mathcal{O}(T_\tau/T_\tau)$ many jobs.

Let $\mathcal{S}$ be the schedule of the task set $\mathbb{T}$ and 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 $\tau_1^c, \tau_2^c, \dots, \tau_{k-1}^c$. More specifically, we have

$$IE = (E_1 \to \tau_1^c \to E_2 \to \tau_2^c \to \dots \to \tau_{k-1}^c \to E_k), \quad (14)$$

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

We utilize the Cutting-Theorem (Theorem 12) to estimate reaction time and data age of IE by its local components, i.e., $\text{Reaction}(IE, \mathcal{S}) \leq \sum_{i=1}^{k} \text{Reaction}(E_i, \mathcal{S}) + \sum_{i=1}^{k-1} \text{Reaction}((\tau_i^c), \mathcal{S})$ and $\text{DataAge}(IE, \mathcal{S}) \leq \sum_{i=1}^{k} \text{DataAge}(E_i, \mathcal{S}) + \sum_{i=1}^{k-1} \text{DataAge}((\tau_i^c), \mathcal{S})$. Please note that $(\tau_i^c),\ i = 1, \dots, k-1$ can be considered as a cause-effect chain of just one task. We apply the bound from Davare et al. [9] to estimate data age and reaction time of (τ_i^c) by $T_{\tau_i^c}^{max} + R_{\tau_i^c}$ under implicit communication.

Corollary 15. *The maximum reaction time and data age of the interconnected cause-effect chain IE under implicit communication can be estimated by the timing behavior of its local parts:*

$$\mathit{Reaction}(IE, \mathcal{S}) \leq \sum_{i=1}^{k} \mathit{Reaction}(E_i, \mathcal{S}) + \sum_{i=1}^{k-1} (T_{\tau_i^c}^{max} + R_{\tau_i^c}) \quad (15)$$

$$\mathit{DataAge}(IE, \mathcal{S}) \leq \sum_{i=1}^{k} \mathit{DataAge}(E_i, \mathcal{S}) + \sum_{i=1}^{k-1} (T_{\tau_i^c}^{max} + R_{\tau_i^c}) \quad (16)$$

Proof: The result follows from the Cutting-Theorem (Theorem 12) and together with the estimation by Davare et al. [9] as discussed above. ■

We note that the values of $\text{Reaction}(E_i, \mathcal{S})$ and $\text{DataAge}(E_i, \mathcal{S})$ can be computed as in Section V-B by fixing the execution time on all ECUs of $E_1, \dots, E_k$.

Under LET we obtain a similar result.

Corollary 16. *The maximum reaction time and data age of the interconnected cause-effect chain IE under LET can be estimated by the timing behavior of its local parts:*

$$\mathit{Reaction}(IE, \mathcal{S}) \leq \sum_{i=1}^{k} \mathit{Reaction}(E_i, \mathcal{S}) + \sum_{i=1}^{k-1} 2T_{\tau_i^c}^{max} \quad (17)$$

$$\mathit{DataAge}(IE, \mathcal{S}) \leq \sum_{i=1}^{k} \mathit{DataAge}(E_i, \mathcal{S}) + \sum_{i=1}^{k-1} 2T_{\tau_i^c}^{max} \quad (18)$$

Proof: This also follows from the Cutting-Theorem. We use that $\text{DataAge}((\tau_i^c), \mathcal{S}) \leq \text{Reaction}((\tau_i^c), \mathcal{S}) \leq 2T_{\tau_i^c}^{max}$ under LET. ■

For Corollary 16 the values of $\text{Reaction}(E_i, \mathcal{S})$ and $\text{DataAge}(E_i, \mathcal{S})$ can be computed by comparing valid augmented job chains with event for external activity during $[0, \Phi + H)$, as noted at the beginning of Section V.

VI. Alternative Data Age Definition

The definition of maximum data age from Dürr et al. [10] differs from our definition of maximum data age in the following way: If we consider the chain of events as outlined in Figure 2, their maximum data age 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 maximum data age from [10], we introduce the definition of a *maximum reduced data age* DataAge*$(E, \mathcal{S})$. It follows Definition 6, except that we set the event for actuation to the processed-event.

Definition 17 (Shortened immediate backward augmented job chain)**.** A *reduced immediate backward augmented job chain* $\overleftarrow{c*}_m^{E,\mathcal{S}}$, $m \in \mathbb{N}$ is the unique augmented job chain $(z, J_1, \dots, J_{|E|}, z')$ where

- the actuation is set to the time of the processed-event, which happens at the m-th write-event of $E(|E|)$, i.e., $z' = we_{J_{|E|}}$ and $J_{|E|} = E(|E|)(m)$,
- the sequence $(J_1, \dots, J_E)$ 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 $\overleftarrow{c*}_m^{E,\mathcal{S}}$ coincide with those of $\overleftarrow{c}_{m+1}^{E,\mathcal{S}}$ and z' is set to $we_{J_{|E|}}$.

We define $\overleftarrow{c*}_m^{E,\mathcal{S}}$ to be *valid* if and only if $\overleftarrow{c}_{m+1}^{E,\mathcal{S}}$ is valid in the sense of Definition 10. The reduced data age is defined similar to Definition 11:

Definition 18 (Maximum reduced data age)**.** For a cause-effect chain E with schedule $\mathcal{S}$ we define the *schedule specific maximum reduced data age* by

$$\text{DataAge*}(E, \mathcal{S}) := \sup\left\{ \ell(\overleftarrow{c*}_m^{E,\mathcal{S}}) \,\middle|\, m \in \mathbb{N}, \overleftarrow{c*}_m^{E,\mathcal{S}} \text{ valid} \right\}, \tag{19}$$

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

The Cutting-Theorem (Theorem 12) is transferred to reduced data age as follows. In the proof of the theorem we cut off an immediate backward augmented job chain at the beginning. This works independent 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 all cause-effect chains $E = (E_1 \to E_2)$ this leads to:

$$\text{DataAge*}(E, \mathcal{S}) \leq \text{DataAge}(E_1, \mathcal{S}) + \text{DataAge*}(E_2, \mathcal{S}) \tag{20}$$

The computation in the local case is similar to Section V-B. With periodic job releases and fixed execution times, the schedule repeats at $\Phi + 2H$ where Φ and H are maximal phase and hyperperiod of tasks on that ECU. It suffices to simulate all reduced immediate forward augmented job chains whose external activity occurs before $\Phi + 2H$ and compute

$$\text{DataAge*}(E, \mathcal{S}) = \max\left\{ \ell(\overleftarrow{c*}_m^{E,\mathcal{S}}) \,\middle|\, \begin{array}{l} m \in \mathbb{N}, \overleftarrow{c*}_m^{E,\mathcal{S}} \text{ valid}, \\ \overleftarrow{c*}_m^{E,\mathcal{S}}(1) < \Phi + 2H \end{array} \right\}. \tag{21}$$

For the interconnected computation, we rely on the new cutting theorem from Eq. (20) and obtain

$$\begin{aligned} \text{DataAge*}(IE, \mathcal{S}) \leq \sum_{i=1}^{k-1} &(\text{DataAge}(E_i, \mathcal{S}) + T_{\tau_i^c}^{max} + R_{\tau_i^c}) \\ &+ \text{DataAge*}(E_k, \mathcal{S}), \end{aligned} \tag{22}$$

as in Corollary 15 for any interconnected cause-effect chain $IE = (E_1 \to \tau_1^c \to E_2 \to \dots \to \tau_{k-1}^c \to E_k)$. The local values of maximum (reduced) data age from Eq. (22) are computed by simulating all (reduced) immediate backward augmented job chains in a bounded time frame and using Eq. (13) and Eq. (21).

VII. 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 in this domain, we evaluate the analyses using synthesized task sets and cause-effect chains that adhere to the details described in *Automotive Benchmarks For Free* [19]. Furthermore, we use task sets that are generated using the UUnifast algorithm [6] to assess the performance for general task parameters. We consider periodic task sets and implicit job communication to apply our analysis results from Section V and Section VI.

Intra-ECU setup: All tasks in each cause-effect chain are mapped to one ECU and use the 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. The evaluation is released on Github [24].

In the following, we use the method by Davare et al. [9] to normalize all other end-to-end bounds, since this method yields the most pessimistic result. We define the *latency reduction* $G(m)$ of an analysis method m with respect to an evaluated bound $B(\cdot)$, e.g., maximum reaction time, by

$$G(m) := (B(davare) - B(m))/B(davare) \cdot 100 \ [\%]. \tag{23}$$

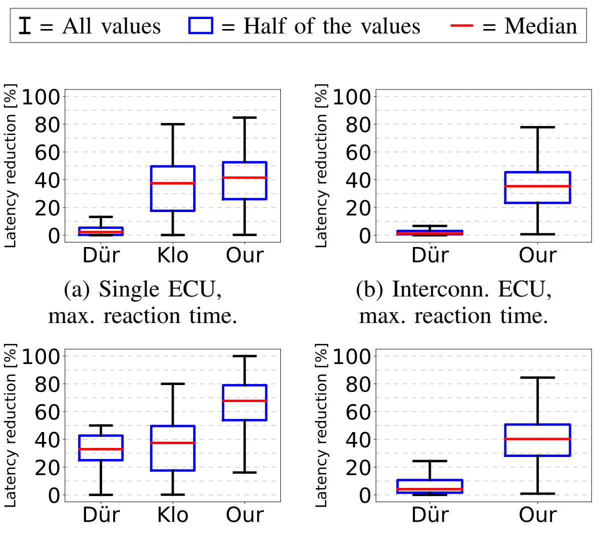
A. Task and Task Set Generation

In this evaluation, a task τ_i is described by the worst-case execution time 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 [19]: Under the automotive benchmark, a task τ_i is generated as follows:

1) The period T_i in ms of a task τ_i is drawn from the set $T = \{1, 2, 5, 10, 20, 50, 100, 200, 1000\}$ according to the related share[2] of [19, Table III, IV and V].

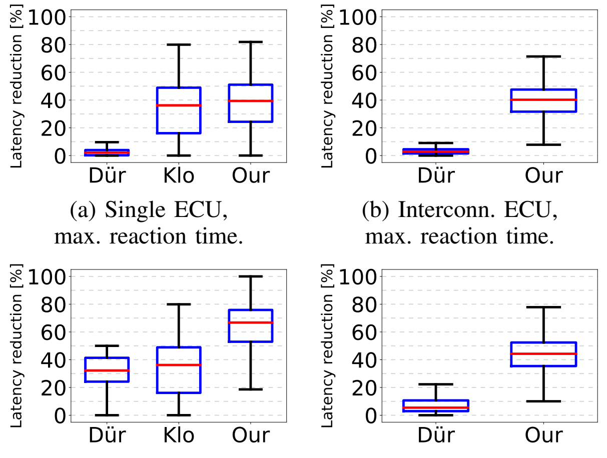
[2]The sum of the probabilities in [19, Table III, IV and V] is only 85%. The remaining 15% is reserved for angle-synchronous tasks that we do not consider. Hence, all share values are divided by 0.85 in the generation process.



(a) Single ECU, max. reaction time. (b) Interconn. ECU, max. reaction time.

(c) Single ECU, max. (reduced) data age. (d) Interconn. ECU, max. (reduced) data age.

Figure 7: Reduction G [%] of the end-to-end latencies compared to Davare's method for the **automotive benchmark**. Our method improves the state-of-the-art for all setups.



(a) Single ECU, max. reaction time. (b) Interconn. ECU, max. reaction time.

(c) Single ECU, max. (reduced) data age. (d) Interconn. ECU, max. (reduced) data age.

Figure 8: Reduction G [%] of the end-to-end latencies compared to Davare's method for the **uniform benchmark**. Again, our method improves the state-of-the-art for all setups.

2) The average-case execution time (ACET) of a task is generated based on a Weibull distribution that fulfills the properties as given in [19, Table III, IV and V].
3) The worst-case execution time (WCET) for a task is determined by drawing a WCET factor, equally distributed from the interval $[f_{min}, f_{max}]$, which is then multiplied with the task's ACET.

For the single ECU case, we generate 1000 *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 worst-case execution-time 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 1000 to 1500 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 [6]: For user-specified values $n \in \mathbb{N}$ and $0 < U^* \leq 1$, the UUniFast algorithm draws utilizations $(U_1, U_2, \dots, U_n)$ 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 1000 task sets with 50 tasks each. Each task's period is drawn from the interval $[1, 2000]$ according to a log-uniform distribution and rounded to the next smallest period in the set $\{1, 2, 5, 10, 20, 50, 100, 200, 500, 1000\}$. Given the periods and utilizations, the worst-case execution-time 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. [17] is formulated only for cause-effect chains with synchronous tasks. Hence, we consider synchronous task sets (with $\phi_i = 0$) in this evaluation.

B. *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 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) would need $C_i = 130 \cdot 10^{-3}$ ms. Given the set of all messages and with random priority assignment, the worst-case response time of each communication task is calculated using time-demand analysis for non-preemptive tasks.

C. *Cause-Effect Chain Generation*

Intra-ECU cause-effect chain generation: Given a generated task set as described in Section VII-A, the set of cause-effect chains is generated according to the description in Section IV-E in [19]. 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 [19].

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 shown in Table VII in [19] 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 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 and priority order under a uniform distribution. For each selection, we create 20 communication tasks as described in Section VII-B (one for each ECU pair). We choose 4 among them to connect the 5 communication tasks.

D. Evaluation Results

In Figure 7 and 8 we show the evaluation results for the automotive and uniform task generation. The boxplots display the improvement of the methods by Dürr et al. [10] **(Dür)** and Kloda et al. [17] **(Klo)**, as well as of *ours-local* in Section V-B and *ours-distributed* **(Our)** in Section V-C over the method by Davare [9] by using the latency reduction G from Eq. (23). Since our definition of *maximum reduced data age* from Section VI coincides with the definition of data age from **Dürr** and **Kloda**, we compare **Our** using maximum reduced data age instead. The method by Schlatow et al. [23] is omitted since it is dominated by Davare's method for non-harmonic task systems. In almost all cases, our methods clearly outperform state-of-the-art analyses. Only for the estimation of maximum reaction time in Figure 7a and 8a in the single ECU scenario, our methods perform similar to Kloda. Especially for the interconnected case, our methods improve the state-of-the-art significantly.

E. 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 2,25GHz and 256GB RAM. Each measurement runs on one independent thread and covers the time for simulation of the schedule and for deriving the single ECU maximum reaction time. Both experiments rely on uniform task generation [6] 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 $\mathrm{lcm}(1, 2, \dots, 20) = 232,792,560$. From each task set, we choose 5 tasks at random without replacement and combine them to a cause-effect chain.

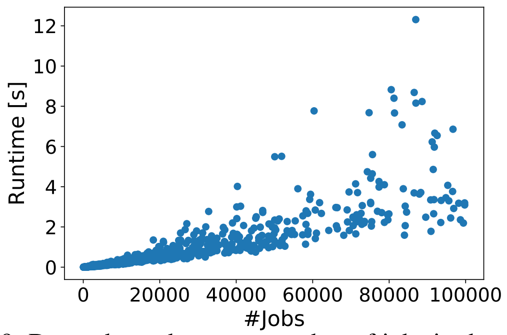


Figure 9: Dependency between number of jobs in the schedule and runtime of our analysis.

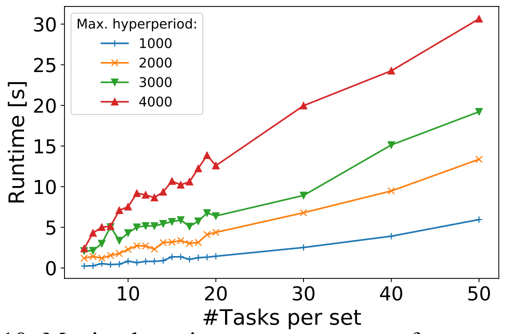


Figure 10: Maximal runtime measurements of our analysis for different hyperperiod bounds.

For 1) we generate $10,000$ task sets with 5 to 20 tasks each. The results are depicted in Figure 9 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 V-B. For fixed number of tasks and 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 maximal runtimes sorted by hyperperiod bounds are depicted in Figure 10. We see that with a low hyperperiod, the maximum runtime can be controlled.

VIII. Conclusion

In this paper, we analyze the *maximum reaction time* and *maximum data age*. We present a precise definition in terms of augmented job chains and an exact local analysis if we fix the execution of all jobs to their worst-case. 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 by partitioning cause-effect chains not only at the ECU-communication but also on one ECU.

Acknowledgment

This work has been supported by European Research Council (ERC) Consolidator Award 2019, as part of PropRT (Number 865170), and by Deutsche Forschungsgemeinschaft (DFG), as part of Sus-Aware (Project no. 398602212)

References

[1] AUTOSAR. Specification of timing extensions, release 4.3.1, Aug. 2017.

[2] M. Becker, D. Dasari, S. Mubeen, M. Behnam, and T. Nolte. Mechaniser-a timing analysis and synthesis tool for multi-rate effect chains with job-level dependencies. In *Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2016.

[3] M. Becker, D. Dasari, S. Mubeen, M. Behnam, and T. Nolte. Synthesizing job-level dependencies for automotive multi-rate effect chains. In *International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 159–169, 2016.

[4] M. Becker, S. Mubeen, D. Dasari, M. Behnam, and T. Nolte. A generic framework facilitating early analysis of data propagation delays in multi-rate systems. In *International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–11, 2017.

[5] A. Benveniste, P. Caspi, P. L. Guernic, H. Marchand, J.-P. Talpin, and S. Tripakis. A protocol for loosely time-triggered architectures. In *EMSOFT*, pages 252–265, 2002.

[6] E. Bini and G. C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005.

[7] Bosch. Controller area network specification 2.0, 1991.

[8] H. Choi, M. Karimi, and H. Kim. Chain-based fixed-priority scheduling of loosely-dependent tasks. In *International Conference on Computer Design (ICCD)*. IEEE, 2020.

[9] A. Davare, Q. Zhu, M. D. Natale, C. Pinello, S. Kanajan, and A. L. Sangiovanni-Vincentelli. Period optimization for hard real-time distributed automotive systems. In *Design Automation Conference, DAC*, pages 278–283, 2007.

[10] M. Dürr, G. von der Brüggen, K.-H. Chen, and J.-J. Chen. End-to-end timing analysis of sporadic cause-effect chains in distributed systems. *ACM Trans. Embedded Comput. Syst. (Special Issue for CASES)*, 18(5s):58:1–58:24, 2019.

[11] R. Ernst, L. Ahrendts, and K. B. Gemlau. System level LET: mastering cause-effect chains in distributed systems. In *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, USA, October 21-23, 2018*, pages 4084–4089. IEEE, 2018.

[12] N. Feiertag, K. Richter, J. Nordlander, and J. Jonsson. A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics. In *Workshop on Compositional Theory and Technology for Real-Time Embedded Systems*, 2009.

[13] FlexRay Consortium. Flexray communications system-protocol specification, 2005.

[14] J. Forget, F. Boniol, and C. Pagetti. Verifying end-to-end real-time constraints on multi-periodic models. In *ETFA*, pages 1–8, 2017.

[15] A. Girault, C. Prevot, S. Quinton, R. Henia, and N. Sordon. Improving and estimating the precision of bounds on the worst-case latency of task chains. *IEEE Trans. on CAD of Integrated Circuits and Systems, (Special Issue for EMSOFT)*, 37(11):2578–2589, 2018.

[16] C. M. Kirsch and A. Sokolova. The logical execution time paradigm. In *Advances in Real-Time Systems*, pages 103–120. Springer, 2012.

[17] T. Kloda, A. Bertout, and Y. Sorel. Latency analysis for data chains of real-time periodic tasks. In *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, pages 360–367, 2018.

[18] A. M. Kordon and N. Tang. Evaluation of the age latency of a real-time communicating system using the LET paradigm. In *ECRTS*, volume 165 of *LIPIcs*, pages 20:1–20:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[19] S. Kramer, D. Ziegenbein, and A. Hamann. Real world automotive benchmark for free. In *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2015.

[20] J. Y.-T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Perform. Eval.*, 2(4):237–250, 1982.

[21] A. Rajeev, S. Mohalik, M. G. Dixit, D. B. Chokshi, and S. Ramesh. Schedulability and end-to-end latency in distributed ecu networks: formal modeling and precise estimation. In *International Conference on Embedded Software*, pages 129–138, 2010.

[22] J. Schlatow and R. Ernst. Response-time analysis for task chains in communicating threads. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 245–254, 2016.

[23] J. Schlatow, M. Möstl, S. Tobuschat, T. Ishigooka, and R. Ernst. Data-age analysis and optimisation for cause-effect chains in automotive control systems. In *IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 1–9, 2018.

[24] TU Dortmund LS12. End-to-end timing analysis. https://github.com/tu-dortmund-ls12-rt/end-to-end, 2021.