End-to-End Timing Analysis of Sporadic Cause-Effect Chains in Distributed Systems

MARCO DÜRR, GEORG VON DER BRÜGGEN, KUAN-HSUN CHEN, and JIAN-JIA CHEN,
TU Dortmund University

A cause-effect chain is used to define the logical order of data dependent tasks, which is independent from the execution order of the jobs of the (periodic/sporadic) tasks. Analyzing the worst-case End-to-End timing behavior, associated to a cause-effect chain, is an important problem in embedded control systems. For example, the detailed timing properties of modern automotive systems are specified in the AUTOSAR Timing Extensions.

In this paper, we present a formal End-to-End timing analysis for distributed systems. We consider the two most important End-to-End timing semantics, i.e., the button-to-action delay (termed as the *maximum reaction time*) and the worst-case data freshness (termed as the *maximum data age*). Our contribution is significant due to the consideration of the sporadic behavior of job activations, whilst the results in the literature have been mostly limited to periodic activations. The proof strategy shows the (previously unexplored) connection between the reaction time (data age, respectively) and immediate forward (backward, respectively) job chains. Our analytical results dominate the state of the art for sporadic task activations in distributed systems and the evaluations show a clear improvement for synthesized task systems as well as for a real world automotive benchmark setting.

CCS Concepts: • **Computer systems organization → Real-time systems**.

Additional Key Words and Phrases: End-to-End Timing Analysis, Distributed Systems, Sporadic Cause-Effect Chains, Embedded Control Systems

ACM Reference Format:
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. 1, 1 (December 2019), 24 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Since timeliness is often required to ensure the stability and correct functionality of software operations in industrial systems, timing properties like End-to-End latencies are specifically important. Especially for safety-critical tasks that need to respond to sensor readings, the desired controlling behavior has to be finished/executed in a certain time interval. To react to an effect triggered by a cause, which can be an external activity or information update, multiple tasks have to be performed sequentially. Therefore, *cause-effect chains* [2, 3, 18] have been used to describe the sequence of steps necessary to complete a cause-effect procedure for a certain functionality. A cause-effect chain is a linear, directed, and acyclic graph, where each node is a task and the edges represent the data

Authors' address: Marco Dürr; Georg von der Brüggen; Kuan-Hsun Chen; Jian-Jia Chen, {first-name}.{last-name}@tu-dortmund.de,TU Dortmund University.

This article appears as part of the ESWEEK-TECS special issue and was presented in the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES) 2019.
XXXX-XXXX/2019/12-ART $15.00
https://doi.org/10.1145/1122445.1122456

dependency among these tasks.[1] 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.

There are two orthogonal approaches to deal with the data dependency described by the cause-effect chains. The *active* approach, e.g., [13, 25, 27], controls the release of the jobs in the subsequent tasks in the chain to make sure that the data are correctly written and read. Specifically, Tindell and Clark [27] combined the worst-case timing analysis on uniprocessor and communication systems for analyzing the end-to-end delay of messages. Alternatively, in the *passive* approach, e.g., [2, 3, 5, 11, 12, 16, 24], the dependency of the tasks described in the cause-effect chain only explains how the *data* are *legally* read (consumed) and written (produced) among the jobs (task instances) of the recurrent tasks in the cause-effect chain. Such recurrent tasks can be defined as time-triggered executions [17], quasi-synchronous time-triggered executions in a distributed setting [5] (i.e., locally time-triggered but globally asynchronous), or recurrent executions that are modeled by classical periodic or sporadic real-time tasks [20, 21]. The dependency of the tasks, defined in a cause-effect chain, is *dependent* (independent, respectively) from how the jobs of the periodic/sporadic tasks are executed and activated in the active (passive, respectively) approach. Therefore, the execution of the active approach is dependent upon the activation of the causes, but the execution of the passive approach is independent. In this paper, we consider the passive approach and sporadic real-time tasks.

When analyzing cause-effect chains, two types of End-to-End latency semantics have been primarily considered in the literature:

- *Reaction Time*: Suppose that an external activity updates a register at time t. What is the maximum time interval length, that starts at time t, needed until this update is processed by each task in the cause-effect chain? The maximum reaction time is the first choice in body electronics to analyze the button to action delay.
- *Data Age*: What is the length of the time interval between the moment the first task in the cause-effect chain reads the data and the moment the last task in the cause-effect chain finishes processing the data? The maximum data age is needed for calculating the delay in control engineering.

To ensure a reasonable worst-case behavior, the maximum reaction time and the maximum data age should be safely analyzed. Towards this, in 2009, Feiertag et al. [11] presented different End-to-End latency semantics for the data propagation based on *forward reachability* and *overwriting* of the data. They defined a *First-to-First* path, starting from the *previous non-overwritten data* until the *first output*. The interval length of the longest First-to-First path is the *maximum reaction time*. The path from the last non-overwritten input until the last output (including possible duplications) is labeled as *Last-to-Last*. The interval length of the longest Last-to-Last path is defined as the *maximum data age*.

The concept of First-to-First and Last-to-Last data propagation semantics has been widely used later in industry and academia, also denoted as First-in-First-out (FIFO) and Last-in-Last-out (LILO), respectively, e.g., in [22, 23]. Specifically, the timing analysis tool SymTA/S [26] from Luxoft offers maximum reaction time and maximum data age analyses. Further explorations of End-to-End analyses include formal verifications, e.g., [24], optimization for latency reduction, e.g., [2, 3], early design analysis, e.g., [4], and language specifications, e.g., [12]. Specifically, Rajeev et al. [24] proposed to generate a formal model based on Calendar Automata, a weaker form of Timed Automata, to verify the maximum data age and the maximum reaction time for periodic tasks.

[1] To be precise, a cause-effect chain is a directed acyclic graph with one source and one sink, i.e., multiple paths may lead from source to sink. This results in a set of linear paths that can be analyzed individually and the timing parameter can be determined by analyzing the resulting values, e.g., by taking the maximum. Hence, we consider linear chains to avoid confusion and an unnecessarily difficult notation.

Kloda et al. [16] proposed a formal method to analyze the maximum reaction time of periodic tasks for multiprocessor platforms. Forget et al. [12] defined a language to specify the End-to-End constraints formally and proposed a technique to verify their satisfaction by considering job-level dependencies of periodic tasks that arrive at the same time. Klaus et al. [15] presented an extension of the Real-Time Systems Compiler (RTSC) that takes data propagation delay into account.

Our Contributions: The known analyses of the maximum data age and maximum reaction time are either heavily based on formal verifications or the upper bound by Davare et al. [9] (e.g., utilized in [2–4]). We consider a distributed system, i.e., multiple communicating Electronic Control Units (ECUs), that executes a given periodic or sporadic task set and provide the following contributions:

- We explain the connection between the reaction time (data age, respectively) and an immediate forward (backward, respectively) job chain (defined in Section 3). The analysis of data propagation in the literature has focused on the data freshness, while our definition is based on the communication semantics. We believe that this is a natural way to define the timing properties of cause-effect chains.
- In Section 5, we consider sporadic real-time tasks with specified minimum and maximum inter-arrival times and analyze the worst-case lengths of the immediate forward and backward job chains of a cause-effect chain, analytically dominating the analysis by Davare et al. [9]. This results in a significant improvement in our extensive comparison based on synthesized systems and a real world automotive benchmark [18].
- Moreover, in Section 6, we show that the derived upper bound on the maximum reaction time is always larger than the derived upper bound on the maximum data age.

2 BACKGROUND AND SYSTEM MODEL

This section provides a detailed description of the system model as well as of cause-effect and job chains. A list of notation is listed and described in Table 1 to facilitate the readability.

2.1 System Model

We assume a set of Electronic Control Units (ECUs) connected via broadcast buses, e.g., Control Area Networks (CANs) [7]. We assume a given partition of a set of recurrently activated tasks $\mathbf{T}$ onto the ECUs, such that each task is statically assigned to one ECU and all task instances, called jobs, are executed on that ECU. Since in the analysis all ECUs can be considered individually, we detail the task model for one individual ECU to not unnecessarily increase the complexity of the explanation and notation. Otherwise, to make the notation precise, all parameters would need an additional index to denote the related ECU.

Individual ECUs: On each ECU, a set Γ of n sporadic (or periodic) tasks is executed , i.e., $\Gamma = \{\tau_1, ...\tau_n\}$. Each task τ_i represents an executable that recurrently releases an infinite number of jobs and is described by the tuple $(C_i, T_i^{min}, T_i^{max})$. The worst-case execution time (WCET) of τ_i, i.e., the longest runtime of the task on the assigned ECU without preemption or interrupt, is denoted as C_i. The minimum and maximum inter-arrival times of task τ_i are represented by T_i^{min} and T_i^{max}, where $0 < T_i^{min} \leq T_i^{max} < \infty$. Hence, if an instance of τ_i is released at time t, the next instance cannot be released before time $t + T_i^{min}$ and not after $t + T_i^{max}$. We call a task sporadic if $T_i^{min} < T_i^{max}$ and periodic if $T_i^{min} = T_i^{max}$. The scheduler decides which job, among the jobs in the ready queue, is executed at each time instant. We assume the tasks to be scheduled under a fixed-priority assignment on each ECU, where the priority of τ_i is higher than the priority of τ_j if $i < j$, $\forall i, j \in \{1, ..., n\}$. The schedule may be either preemptive or non-preemptive and the resulting exact schedule is termed S. The ℓ^{th} job of τ_i is denoted by $J_{i,\ell}$. The worst-case response time (WCRT) of τ_i, i.e., the longest time interval between arrival and finishing time of all task

Notation	Description
Γ	Task set of a single ECU
$\tau_i = (C_i, T_i^{min}, T_i^{max})$	Task τ_i, related WCET C_i, and minimum/maximum inter-arrival time $T_i^{min/max}$
$\mathbf{T}$	Task set of all ECUs
R_i	Worst-case response time of τ_i
B_i	Maximum time τ_i can be blocked by the execution of lower prioritized tasks
S	Indicates a specific schedule
E_j	A specific cause-effect chain
K_j	Returns the number of tasks in E_j, i.e., the index of the last task in the cause-effect chain
$J_{i,\ell}$	The ℓ^{th} job of τ_i, where a job is an instance of a task
$a_i^{j,S,\ell}$	The arrival time of the ℓ-th job of the i-th task in the cause-effect chain E_j, i.e., $J_{E_j(i),\ell}$, in schedule S
$\delta_i^{j,S,\ell}$	The starting time of the ℓ-th job of the i-th task in the cause-effect chain E_j, i.e., $J_{E_j(i),\ell}$, in schedule S
$f_i^{jS,\ell}$	The finishing time of the ℓ-th job of the i-th task in the cause-effect chain E_j, i.e., $J_{E_j(i),\ell}$, in schedule S
$\overrightarrow{\varsigma^{j,S,\ell}}$	Forward job chain, related to E_j in schedule S, starting from job ℓ of $\tau_{E_j(1)}$
$\overleftarrow{\varsigma^{j,S,\ell}}$	Backward job chain, related to E_j in schedule S, ending at job ℓ of $\tau_{E_j(K_j)}$

Table 1. Notation used in this work

instances, is represented as R_i. For partitioned multiprocessor scheduling, the WCRT of a task only depends on the ECU it is assigned to and not on jobs that are executed on another ECU.

Uniprocessor WCRT Analysis: Under fixed-priority preemptive scheduling, the WCRT of a task can be calculated using Time Demand Analysis (TDA) by Lehoczky et al. [19]. According to TDA, the WCRT of a task τ_i is the minimum positive value of R_i for which the following equation holds:

$$R_i = C_i + \sum_{\tau_k \in hp(i)} \left\lceil \frac{R_i}{T_k^{min}} \right\rceil C_k \tag{1}$$

where $hp(i)$ is the set of tasks whose priorities are higher than τ_i.

TDA can be extended to the non-preemptive case by including the maximum blocking time B_i, defined as $max_{\tau_k \in lp(i)}\{C_k\}$, where $lp(i)$ is the set of tasks whose priorities are lower than τ_i, resulting in the following sufficient test [8]:

$$R_i = C_i + B_i + \sum_{\tau_k \in hp(i)} \left\lceil \frac{R_i}{T_k^{min}} \right\rceil C_k. \tag{2}$$

We note that the analysis in Eq. (2) is slightly pessimistic and that other forms of WCRT analysis can also be applied. Details can be found for instance in [28].

Note that Eq. (1) and Eq. (2) are only valid to calculate the WCRT if it is smaller than the tasks deadline, which in our case is T_i^{min}. *For the rest of this paper, we assume that for every task τ_i in $\mathbf{T}$, its WCRT is no more than T_i^{min}.*

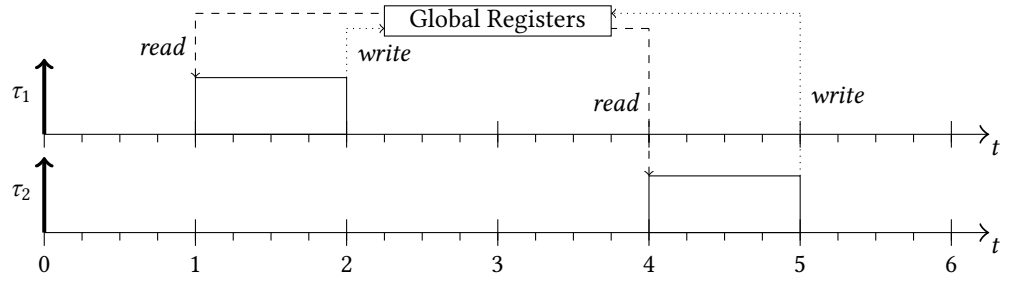


Fig. 1. A visualization of implicit task communication as used in this work, assuming tasks on the same ECU.

Communication Semantics: Since our goal is to analyze the timing behavior among multiple data dependent jobs, knowledge about the specific communication pattern is necessary. In automotive systems, data can be passed among jobs via direct communication (without consistency check, i.e., at a risk of data inconsistency), implicit communication (read at the begin and write at the end of the job), as shown in Figure 1, or the logical execution time model (read and write at predefined time points, e.g., release time of a job released by a periodic task) [14]. Our proposed approach can be adjusted to cover each introduced communication semantic by considering the specific read and write operation times. To simplify the presentation, we focus on implicit communication, where the data is always read at the beginning and written at the end of a job.

2.2 Cause-Effect Chains

A set of *cause-effect chains* Π describes the causal order for the execution of data dependent tasks. One specific cause-effect chain of the set Π is denoted as E_j, i.e., the data flow from one specific start to one specific end task. It can be described as a linear, directed, and acyclic graph (DAG), where each node is a task and edges represent data dependency among tasks. We denote K_j as the number of tasks in E_j, where $K_j \geq 2$. For simplicity of presentation, we assume that no two tasks in the chain are the same.[2] The function $E_j(l)$ returns the index of the l^{th} task of the cause-effect chain E_j. For example, let $E_1 = (\tau_4 \rightarrow \tau_3 \rightarrow \tau_5)$, then $E_1(1) = 4$, $E_1(2) = 3$, and $E_1(3) = 5$. Cause-effect chains are inspired by event-chains of the AUTOSAR Timing Extensions [1], which represent chains of more general functional dependency.

For a chain E_j, two subsequent tasks may either be located on the same or on different ECUs. We assume that two tasks that are assigned to the same ECU communicate directly via shared memory/registers. If two tasks are assigned to different ECUs, they communicate via one of the broadcast buses. For convenience, in our description from now on and in the analysis, we assume the system has only one broadcast bus. Otherwise, the analysis can easily be extended as long as it is clear which bus is used for the communication between two given tasks on different ECUs.

If two subsequent tasks in a chain E_j are located on different ECUs, we model the communication over the broadcast bus as a recurrent communication task on the bus, which is inserted into the cause-effect chain at the point where the communication happens. For instance, for a chain $\tau_1 \rightarrow \tau_2$ where τ_1 and τ_2 are on different ECUs, we insert a communication task τ_c, resulting in the cause-effect chain $\tau_1 \rightarrow \tau_c \rightarrow \tau_2$. The exact characteristics of this communication task depends on the broadcast bus. We abstract this communication by taking the following assumptions: 1) τ_c is released either sporadically or periodically and transfers the needed data and possibly some additional information, e.g., multiple values destined for a set of tasks may be transferred in one

[2]Otherwise, directly consecutive tasks have to be different, and the iterations of a loop must be bounded to form a cause-effect chain.

message, 2) When a job of τ_1 finishes, it writes the necessary values into a buffer similar to the communication in one core and each job of τ_c reads the current value when it is initialized, and 3) After a job of τ_c is finished, τ_2 can directly read the updated value in a similar way as it would read a value that was produced by a task on the same processor.

These abstractions are taken to ease the presentation of the analysis, where only the minimum and maximum inter-arrival times and the WCRT of the communication task are needed. However, these assumptions can usually be adjusted to reflect the communication behavior of the actual system while still maintaining the possibility to be analyzed by our approach. For instance, if the communication task τ_c is activated by τ_1, the inter-arrival time of τ_c depends directly on the worst-case and best-case response time of τ_1 as well as on the minimum and maximum inter-arrival time of τ_1. Furthermore, note that similar assumptions are usually taken in other works that consider End-to-End timing analysis in multiprocessor scenarios.

After all communication tasks are inserted, we implicitly assume that K_j and the indexes of the individual tasks, including the communication tasks, is adjusted to reflect the updated order. We consider fixed-priority non-preemptive scheduling on the broadcast bus, which is the case for a CAN bus. If all communication tasks and their priorities are known, the WCRT of a communication task on a CAN bus can be calculated according to Eq. (2). Otherwise, suitable techniques to calculate the WCRT can be applied.

2.3 Sporadic Task Scenario

When considering cause-effect chains in embedded control systems, it seems reasonable to assume either periodically released task instances or releases based on events created by previous jobs in the cause-effect chain, i.e., a job of a task is released when its predecessor in the cause-effect chain finishes its execution. Nevertheless, if the releases are triggered by a cause that is not part of the chain itself, tasks may release their instances sporadically with respect to a specific cause-effect chain. On one hand, the trigger frequency may result from an external parameter that changes over time or dynamically, e.g., angle-synchronous tasks that are released based on the rotation of the crankshaft [18]. On the other hand, a task may be part of multiple cause-effect chains. In this situation, a task appears to be sporadic (from the perspective of the chain under analysis) if its instances are released based on the finishing time of a job in another chain. We note, that our analysis can directly be applied to (over-)approximate the worst-case End-to-End timing behavior for periodic tasks since periodic task sets are a special case of sporadic task sets where $T_i^{min} = T_i^{max}$ for all tasks.

3 JOB CHAINS

A *job chain* of E_j is a sequence of data dependent jobs for a specific schedule S of the given periodic/sporadic tasks in $\mathbf{T}$. Note that tasks may be located on different ECUs and that we inserted communication tasks, i.e., S covers all ECUs as well as the broadcast bus. However, the schedules on the individual ECUs and on the broadcast bus are independent from each other. We consider two types of job chains, i.e., one defined in a forward manner and one defined in a backward manner, based on the implicit-communication semantics, S, and E_j.

For the rest of the paper, we analyze each of the cause-effect chains in the systems one by one. For simplicity of notation, we (in most places) implicitly drop the indexes related to cause-effect chain j, schedule S, and ℓ^{th} job when the context is clear.

3.1 Immediate forward job chain

An immediate forward job chain that starts from the ℓ-th job of the first task in the cause-effect chain E_j, is denoted as $\overrightarrow{\varsigma^{j,S,\ell}}$, i.e., it starts at $J_{E_j(1),\ell}$. In the following, we define the immediate forward job chain $\overrightarrow{\varsigma^{j,S,\ell}}$ iteratively from $J_{E_j(1),\ell}$:

(1) The first job of the immediate forward job chain is by definition $J_{E_j(1),\ell}$. For the job $J_{E_j(1),\ell}$, we define $a_1^{j,S,\ell}$ as its arrival time, $\delta_1^{j,S,\ell}$ as its starting time, and $f_1^{j,S,\ell}$ as its finishing time in the schedule S.

(2) For each $i = 2, 3, \ldots, K_j$, we define the i^{th} job in the immediate forward job chain iteratively. Let $J_{E_j(i),\sharp}$ be the *first* job of $\tau_{E_j(i)}$ that starts its execution no earlier than $f_{i-1}^{j,S,\ell}$ in the schedule S. This job $J_{E_j(i),\sharp}$ is by definition the first job in the schedule S, which uses the processing result in the cause-effect chain E_j. Therefore, $J_{E_j(i),\sharp}$ is the i^{th} job in the immediate forward job chain $\overrightarrow{\varsigma^{j,S,\ell}}$. For the job $J_{E_j(i),\sharp}$, we define $a_i^{j,S,\ell}$ as its arrival time, $\delta_i^{j,S,\ell}$ as its starting time, and $f_i^{j,S,\ell}$ as its finishing time in the schedule S.

(3) The length of the immediate forward job chain $\overrightarrow{\varsigma^{j,S,\ell}}$ is defined as $f_{K_j}^{j,S,\ell} - a_1^{j,S,\ell}$.

Definition 3.1 (worst-case forward job chain). The worst-case length of the immediate forward job chains ($WCFC_j$) of a cause-effect chain E_j is defined by considering all possible schedules S and all possible immediate forward job chains, i.e.,

$$WCFC_j = \max_{S} \max_{\overrightarrow{\varsigma^{j,S,\ell}}, \forall \ell = 1,2,\ldots} f_{K_j}^{j,S,\ell} - a_1^{j,S,\ell} \tag{3}$$

3.2 Immediate backward job chain

An immediate backward job chain that ends at the ℓ-th job of the last task in the cause-effect chain E_j, is denoted as $\overleftarrow{\varsigma^{j,S,\ell}}$, i.e., it ends at $J_{E_j(K_j),\ell}$. In the following, we define the immediate backward job chain $\overleftarrow{\varsigma^{j,S,\ell}}$ iteratively from $J_{E_j(K_j),\ell}$:

(1) The last job of the immediate backward job chain is by definition $J_{E_j(K_j),\ell}$. For the job $J_{E_j(K_j),\ell}$, we define $a_{K_j}^{j,S,\ell}$ as its arrival time, $\delta_{K_j}^{j,S,\ell}$ as its starting time, and $f_{K_j}^{j,S,\ell}$ as its finishing time in the schedule S.

(2) For each $i = K_j - 1, K_j - 2, \ldots, 1$, we define the i^{th} job in the immediate backward job chain iteratively. Let $J_{E_j(i),\flat}$ be the *last* job of $\tau_{E_j(i)}$ that finishes its execution no later than the starting time $\delta_{i+1}^{j,S,\ell}$ of the next job in the backward job chain in the schedule S. This job $J_{E_j(i),\flat}$ is by definition the last job in the schedule S, which offers the processing result to the next job of the cause-effect chain E_j. Therefore, $J_{E_j(i),\flat}$ is the i^{th} job in the immediate backward job chain $\overleftarrow{\varsigma^{j,S,\ell}}$. For the job $J_{E_j(i),\flat}$, we define $a_i^{j,S,\ell}$ as its arrival time, $\delta_i^{j,S,\ell}$ as its starting time, and $f_i^{j,S,\ell}$ as its finishing time in the schedule S.

(3) The length of the immediate backward job chain $\overleftarrow{\varsigma^{j,S,\ell}}$ is defined as $f_{K_j}^{j,S,\ell} - a_1^{j,S,\ell}$. It is possible that there is no immediate backward job chain ending with the job $J_{E_j(K_j),\ell}$, i.e., $a_1^{j,S,\ell}$ cannot be defined. Since we are interested in the worst-case length, we simply set $a_1^{j,S,\ell}$ to $f_{K_j}^{j,S,\ell}$, i.e., the length of the job chain is set to 0.

Definition 3.2 (worst-case backward job chain). The worst-case length of the immediate backward job chains ($WCBC_j$) of an cause-effect chain E_j is defined by considering all possible schedules S

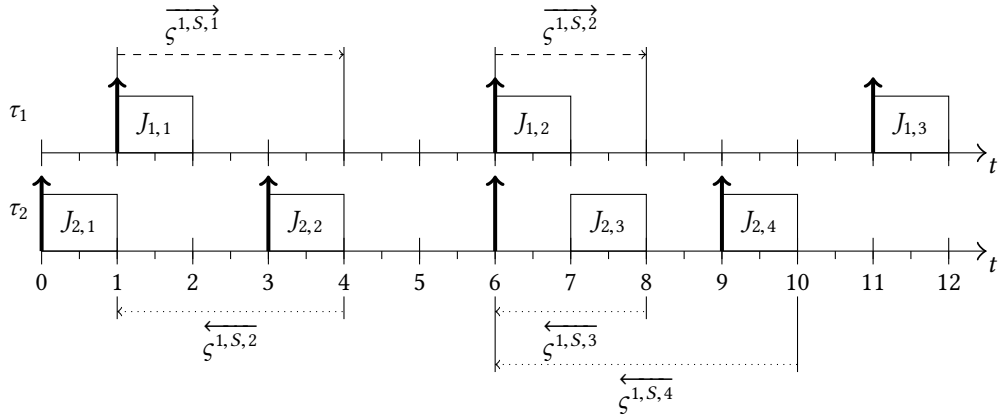


Fig. 2. The presented Schedule S illustrates an example of forward and backward job chains. The notation of a chain $\varsigma^{j,S,\ell}$ denotes cause-effect chain E_j in schedule S, starting at job $J_{i,\ell}$, i.e., the ℓ^{th} job of task $\tau_{E_j(1)}$.

and all possible immediate backward job chains, i.e.,

$$WCBC_j = \max_{S} \max_{\overleftarrow{\varsigma^{j,S,\ell}}, \forall \ell = 1,2,\dots} f_{K_j}^{j,S,\ell} - a_1^{j,S,\ell} \tag{4}$$

3.3 Examples for Job Chains

An example of immediate forward and backward job chains is depicted in Figure 2. The shown schedule S consists of two tasks: $\tau_1 = (1, 5, 5)$ and $\tau_2 = (1, 3, 3)$. Suppose that a cause-effect chain $E_1 = (\tau_1 \rightarrow \tau_2)$ is under analysis, such that it shows two immediate forward job chains $\overrightarrow{\varsigma^{1,S,1}} = (J_{1,1}, J_{2,2})$ and $\overrightarrow{\varsigma^{1,S,2}} = (J_{1,2}, J_{2,3})$. Furthermore, there are three immediate backward job chains, to be precise $\overleftarrow{\varsigma^{1,S,2}} = (J_{1,1}, J_{2,2})$, $\overleftarrow{\varsigma^{1,S,3}} = (J_{1,2}, J_{2,3})$, and $\overleftarrow{\varsigma^{1,S,4}} = (J_{1,2}, J_{2,4})$. For this schedule, it is not possible to construct an immediate backward job chain that ends at job $J_{2,1}$. This means that job $J_{2,1}$ does not produce any meaningful output for the cause-effect chain E_1 and is therefore neglected.

4 PROBLEM DEFINITION: END-TO-END LATENCY SEMANTICS

We analyze the worst-case time interval from a cause to an effect, i.e., the time interval from the moment where the first task in a cause-effect chain starts executing until the point in time where the last task in a cause-effect chain finishes. An End-to-End latency analysis of a cause-effect chain is necessary to guarantee an upper bound on the worst-case time intervals and to verify the End-to-End timing requirements of the underlying procedure. Two End-to-End latency semantics are of specific interest: the maximum data age and the maximum reaction time as introduced by Feiertag et al. [11].

The *maximum data age*, or Last-to-Last, is the time interval between the moment a task starts to sample a value until the last point in time the system produces an output related to that sample. Hence, this semantic is of great concern to control engineers. The length of the worst-case immediate backward job chain $WCBC_j$ (Definition 3.2) corresponds to the maximum data age.

Definition 4.1 (maximum data age). The maximum data age of a cause-effect chain E_j in schedule S is less or equal to the worst-case immediate backward job chain length $WCBC_j$.

In body electronics the *maximum reaction time*, also called First-to-First, is of interest. This semantic refers to the first response of the system to an external cause, e.g., a button press or a value change of a register. Thus, the time interval between the worst-case occurrence of a cause and the first corresponding output of the system needs to be analyzed. To determine the maximum reaction time of a cause-effect chain, the worst-case intermediate forward job chain length is extended by the maximum inter-arrival time of the first task in the cause-effect chain. This extension is necessary, since a cause can arrive immediately after a job starts its execution, such that the cause is processed by the next job of the task.

Definition 4.2 (maximum reaction time). The reaction time of a cause-effect chain E_j in schedule S is less or equal to $WCFC_j + T^{max}_{E_j(1)}$.

Figure 3 depicts an example of various End-to-End semantics. The shown exact schedule S includes three tasks, τ_1, τ_2, and τ_3, with the following properties: $C_i = 0.5\ \forall \tau_i$, $T^{min}_{1,3} = T^{max}_{1,3} = 2$, and $T^{min}_{2} = T^{max}_{2} = 6$. The formulations of the presented time intervals at the bottom refer to overwriting effects, e.g., Last-to-Last describes the time interval between the last non-overwritten input to the last produced output, and First-to-First is related to the previous non-overwritten input to the first produced output. Furthermore, the worst-case forward and backward job chains, $\overrightarrow{\varsigma^{1,S,2}}$ and $\overleftarrow{\varsigma^{1,S,3}}$, are marked at the top. The maximum data age semantic corresponds with the worst-case backward job chain $\overleftarrow{\varsigma^{1,S,3}}$ by definition. The worst-case forward job chain $\overrightarrow{\varsigma^{1,S,2}}$ represents the time interval between the first job of the first task in a cause-effect chain that is overwritten to the first corresponding output. Hence, extending the worst-case forward job chain by the maximal inter-arrival time of the first task in the cause-effect chain results in a time interval between the previous non-overwritten job to the first related output that is defined as the maximum reaction time.

Davare et al. [9] provided an upper bound on both the maximum data age and the maximum reaction time by summing up the worst-case response times and the periods of all the tasks in the cause-effect chain, as shown in Eq. (5).

$$bound_{Davare} = \sum_{i=1}^{K_j} T^{max}_{E_j(i)} + R_{E_j(i)} \tag{5}$$

We note that the $bound_{Davare}$ is probably trivial but it has been never formally proved. This worst-case happens when (1) a job of a periodic task $\tau_{E_j(i+1)}$ finishes as early as possible but the data is updated/written by task $\tau_{E_j(i)}$ right after the job started, and (2) then the subsequent job of $\tau_{E_j(i+1)}$ suffers from its WCRT. This way of analyzing the End-to-End delay is pessimistic since a task is allowed to release two consecutive jobs in the worst imaginable setup. However, this may not be possible if the tasks in the task set are strictly periodic, and if they have fixed release times of their first jobs (also called phases or offsets).

Table 2 shows the results of End-to-End delay analyses related to the example of Figure 3 for the maximum reaction time and maximum data age. We compare $bound_{Davare}$ with the proposed analyses $bound_{ours}$ in Section 5 and the exact values as stated in the figure. It shows a significant gap between $bound_{Davare}$ and $bound_{ours}$.

To the best of our knowledge, no formal End-to-End timing analysis exists that distinguishes between maximum data age and maximum reaction time for sporadic tasks. In the following sections, we exploit the formerly defined job chains to explicitly analyze these two End-to-End semantics. Furthermore, the correctness of the analyses approaches are proved, such that a comprehensible End-to-End analysis can be performed.

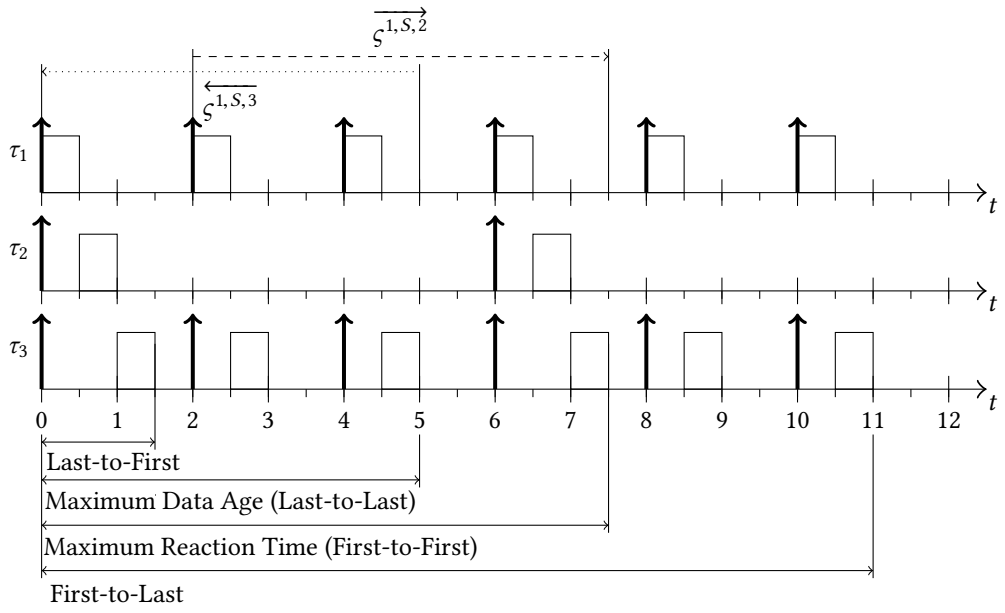


Fig. 3. Data propagation path of different semantics.

5 ANALYSIS FOR CAUSE-EFFECT CHAINS OF SPORADIC TASK SYSTEMS

In this section, we explain how the maximum reaction time and the maximum data age can be upper bounded for multiple sporadic task systems. This is done by considering job chains that result from the cause-effect chain E_j. Please note that, for the rest of this section, $a_i^{j,S,\ell}$, $\delta_i^{j,S,\ell}$ and $f_i^{j,S,\ell}$ respectively stand for the arrival, the starting, and the finishing time of the job $J_{E_j(i),\sharp}$, which is the i^{th} job in the cause-effect chain $E_j(i)$.

5.1 Maximum Reaction Time

An upper bound of the maximum reaction time is derived by looking at any two consecutive tasks in a cause-effect chain, where the gap between the release times of the related jobs are bounded in any possible forward job chain. We start by looking at one specific ECU and extend our result upon multiple communicating ECUs. To facilitate the comprehension of the presented proofs, a partial schedule is visualized in Figure 4.

Lemma 5.1. *Suppose* Γ *to be executed on one ECU and that* R_k *of task* τ_k *is no more than* T_k^{min} *for every task* τ_k *in* Γ *under uniprocessor preemptive / non-preemptive fixed-priority scheduling. Then, for any resulting schedule* S, $\ell = 1, 2, \ldots$, *and* $i = 1, 2, \ldots, K_j - 1$,

$$a_{i+1}^{j,S,\ell} - a_i^{j,S,\ell} \leq \max\left\{R_{E_j(i)}, T_{E_j(i+1)}^{max} + R_{E_j(i)} \cdot [E_j(i) > E_j(i+1)]\right\} \tag{6}$$

where $[E_j(i) > E_j(i+1)]$ *is the Iverson bracket, which is* 1 *when the* $(i+1)^{th}$ *task in the cause-effect chain* E_j *has a higher priority than the* i^{th} *task in the cause-effect chain and* 0 *otherwise.*

Lemma 5.1 allows the tasks in the cause-effect chain to have arbitrary periods. Specifically, both over- and undersampling are considered and for two consecutive tasks in the cause-effect chain the second task may have a higher priority than the first task.

	Maximum Reaction Time	Maximum Data Age
$bound_{Davare}$	13	13
$bound_{ours}$	9.5 (Theorem 5.4)	7.5 (Theorem 5.10)
Exact	7.5	5

Table 2. Comparison of End-to-End delay analyses

Proof. Let $J_{E_j(i),p}$ and $J_{E_j(i+1),q}$ be the i^{th} and $(i+1)^{th}$ job in the immediate forward job chain $\overrightarrow{\varsigma^{j,S,\ell}}$ that are executed on the same ECU. There are two scenarios to consider: (1) $J_{E_j(i+1),q}$ arrives *before* $J_{E_j(i),p}$ finishes in S, and (2) $J_{E_j(i+1),q}$ arrives *not before* $J_{E_j(i),p}$ finishes in S. Since j, S, and ℓ are fixed in the statement of the lemma, we drop these indexes for the simplicity of presentation, i.e., a_i and f_i stand for $a_i^{j,S,\ell}$ and $f_i^{j,S,\ell}$, respectively.

- **Case 1** $a_{i+1} < f_i$, i.e., $J_{E_j(i+1),q}$ arrives before $J_{E_j(i),p}$ finishes. Since the worst-case response time of $J_{E_j(i),p}$ is at most $R_{E_j(i)} \leq T^{min}_{E_j(i)}$ and $a_{i+1} < f_i$, we know that

$$a_{i+1} - a_i < f_i - a_i \leq R_{E_j(i)} \tag{7}$$

- **Case 2** $a_{i+1} \geq f_i$, i.e., $J_{E_j(i+1),q}$ arrives at or after $J_{E_j(i),p}$ finishes. By our assumption that $R_{E_j(i+1)} \leq T^{min}_{E_j(i+1)}$, there must be another job $J_{E_j(i+1),q-1}$ that arrived prior to job $J_{E_j(i+1),q-1}$, i.e., job $J_{E_j(i+1),q-1}$ exists. By the definition of an immediate forward job chain, job $J_{E_j(i+1),q-1}$ cannot arrive at or after $J_{E_j(i),p}$ finishes; otherwise job $J_{E_j(i+1),q-1}$ should be in the immediate forward job chain $\overrightarrow{\varsigma^{j,S,\ell}}$ instead of job $J_{E_j(i+1),q}$. Let r be the arrival time of job $J_{E_j(i+1),q-1}$. By the above discussion, $r < f_i$. Moreover, by the definition of the maximum inter-arrival time of a task, we know that $r \geq a_{i+1} - T^{max}_{E_j(i+1)}$. By these two inequalities, we have

$$a_{i+1} - a_i = (a_{i+1} - r) + (r - a_i) \leq (a_{i+1} - r) + (f_i - a_i) \leq T^{max}_{E_j(i+1)} + R_{E_j(i)} \tag{8}$$

  The inequality in Eq. (8) can be improved when $E_j(i) < E_j(i+1)$, i.e., the priority of task $\tau_{E_j(i)}$ is higher than task $\tau_{E_j(i+1)}$. If this holds, the arrival time r of job $J_{E_j(i+1),q-1}$ must be less than the arrival time of job $J_{E_j(i),p}$, i.e., $r < a_i$. Otherwise, the starting time of $J_{E_j(i+1),q-1}$ in the schedule S must be after f_i according to the preemptive fixed-priority scheduling strategy, which contradicts to the definition of the immediate forward job chain. Therefore, by $r < a_i$ and $r \geq a_{i+1} - T^{max}_{E_j(i+1)}$, we have

$$a_{i+1} - a_i \leq a_{i+1} - r \leq T^{max}_{E_j(i+1)} \text{ if } E_j(i) < E_j(i+1) \tag{9}$$

By combining the above cases, we reach the inequality in Eq. (6). □

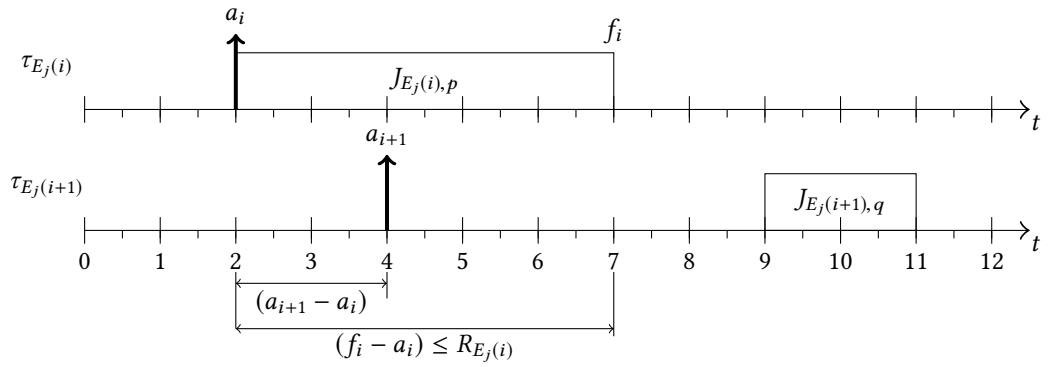
Lemma 5.2. *Let $J_{E_j(i),p}$ and $J_{E_j(i+1),q}$ be the i^{th} and $(i+1)^{th}$ jobs in the immediate forward job chain $\overrightarrow{\varsigma^{j,S,\ell}}$ that are executed on different ECUs. Then, Eq.* (6) *needs to be extended, such that:*

$$a_{i+1}^{j,S,\ell} - a_i^{j,S,\ell} \leq \max\left\{R_{E_j(i)}, T^{max}_{E_j(i+1)} + R_{E_j(i)} \cdot [P]\right\} \tag{10}$$

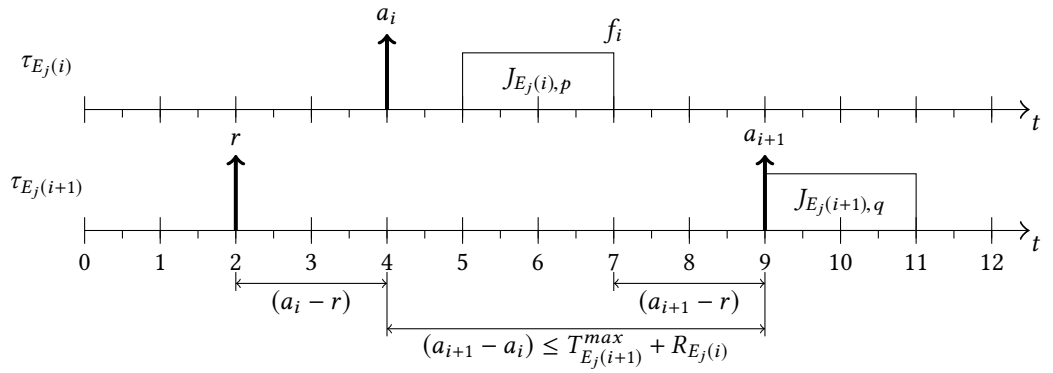
where $[P]$ is the Iverson bracket, with the following proposition:

$$[P] = \begin{cases} 1 & \text{if } E_j(i) > E_j(i+1) \vee ECU(E_j(i)) \neq ECU(E_j(i+1)) \\ 0 & \text{otherwise} \end{cases}$$

In other words, the Iverson bracket $[P]$ is 1 when the $(i+1)^{th}$ task in the cause-effect chain E_j has a higher priority or is executed on a different ECU than the i^{th} task in the cause-effect chain, 0 otherwise.



(a) Case 1: $a_{i+1} < f_i$, i.e., $J_{E_j(i+1),q}$ arrives before $J_{E_j(i),p}$ finishes.



(b) Case 2: $a_{i+1} \geq f_i$, i.e., $J_{E_j(i+1),q}$ arrives at or after $J_{E_j(i),p}$ finishes.

Fig. 4. Visualization of the two considered scenarios in the maximum reaction time delay analysis.

Proof. This follows from Lemma 5.1, since Eqs (7) and (8) hold. The improvement in Eq. (9) can not be applied and is therefore extended, such that we reach the inequality in Eq. (10). □

This allows to bound the length of $WCFC_j$.

Theorem 5.3. *Suppose that R_k of task τ_k is no more than T^{min}_k for every task τ_k in Γ. For a cause-effect chain E_j*

$$WCFC_j \leq R_{E(K_j)} + \sum_{i=1}^{K_j - 1} \max \left\{ R_{E_j(i)}, T^{max}_{E_j(i+1)} + R_{E_j(i)} \cdot [P] \right\} \tag{11}$$

Proof. By definition

$$
\begin{aligned}
WCFC_j &= \max_S \max_{\overrightarrow{\varsigma^{j,S,\ell}}, \forall \ell = 1,2,\ldots} f_{K_j}^{j,S,\ell} - a_1^{j,S,\ell} \\
&= \max_S \max_{\overrightarrow{\varsigma^{j,S,\ell}}, \forall \ell = 1,2,\ldots} f_{K_j}^{j,S,\ell} - a_{K_j}^{j,S,\ell} + \left(\sum_{i=1}^{K_j - 1} a_{i+1}^{j,S,\ell} - a_i^{j,S,\ell} \right) \\
&\leq R_{E(K_j)} + \sum_{i=1}^{K_j - 1} \max \left\{ R_{E_j(i)}, T_{E_j(i+1)}^{max} + R_{E_j(i)} \cdot [P] \right\} \qquad (12)
\end{aligned}
$$

where $\leq$ is due to Lemma 5.2. □

This leads to an upper bound of the maximum reaction time.

Theorem 5.4. *Suppose that the worst-case response time R_k of task τ_k is no more than T_k^{min} for every task τ_k in Γ. The maximum reaction time of a cause-effect chain E_j is upper bounded by $T_{E_j(1)}^{max} + R_{E(K_j)} + \sum_{i=1}^{K_j - 1} \max \left\{ R_{E_j(i)}, T_{E_j(i+1)}^{max} + R_{E_j(i)} \cdot [P] \right\}$*

Proof. This is due to Definition 4.2, Theorem 5.3, and the observation that the interval between the cause and the moment the job of $\tau_{E_j(1)}$ that starts the immediate forward job chain is released is bounded by $T_{E_j(1)}^{max}$. □

Note that the Iverson bracket is always 0 if the tasks in the cause-effect chain appear in decreasing order of priority and are executed on the same ECU, which directly gives the following Corollary.

Corollary 5.5. *Suppose that the worst-case response time R_k of task τ_k is no more than T_k^{min} for every task τ_k in Γ. The maximum reaction time of a cause-effect chain E_j is upper bounded by $T_{E_j(1)}^{max} + R_{E(K_j)} + \sum_{i=1}^{K_j - 1} \max \left\{ R_{E_j(i)}, T_{E_j(i+1)}^{max} \right\}$ if the tasks in the cause-effect are in decreasing order of priority and executed on the same ECU.*

The opposite side of the above corollary when the Iverson bracket is always 1 is in fact the analysis in Eq. (5) by Davare et al. [9].

Theorem 5.6. *The analysis in Theorem 5.4 analytically dominates the analysis in Eq. (5) by Davare et al. [9].*

Proof. When $[P]$ is always 1 for $j = 1, 2, \ldots, K_j - 1$, the upper bound in Theorem 5.4 is exactly the right hand side of Eq. (5). Since this is the worst case in our analysis in Theorem 5.4, our analysis anlytically dominates Eq. (5). □

5.2 Maximum Data Age

After determining the worst-case reaction time, we now look at the maximum data age, which is bounded based on the immediate backward job chains. Again, we start with the case that the tasks are on the same ECU. To facilitate the comprehension of the presented proofs, a partial schedule is visualized in Figure 5.

Lemma 5.7. *Suppose Γ to be executed on one ECU and that R_k of task τ_k is no more than T_k^{min} for every task τ_k in Γ under uniprocessor preemptive / non-preemptive fixed-priority scheduling. If the immediate backward job chain $\overleftarrow{\varsigma^{j,S,\ell}}$ can be defined with K_j jobs, then for any uniprocessor preemptive / non-preemptive fixed-priority schedule S, $\ell = 1, 2, \ldots$, and $i = 1, 2, \ldots, K_j - 1$ holds:*

$$
a_{i+1}^{j,S,\ell} - a_i^{j,S,\ell} \leq T_{E_j(i)}^{max} + R_{E_j(i)} \cdot [E_j(i) > E_j(i+1)] \qquad (13)
$$

where $[E_j(i) > E_j(i+1)]$ *is the Iverson bracket.*

Proof. The proof is similar to that of Lemma 5.1, but in the opposite direction. We drop j, S, and ℓ for the simplicity of presentation, i.e., a_i, δ_i, and f_i stand for $a_i^{j,S,\ell}$, $\delta_i^{j,S,\ell}$, and $f_i^{j,S,\ell}$, respectively. Suppose that $J_{E_j(i),p}$ and $J_{E_j(i+1),q}$ are the i^{th} and $(i+1)^{th}$ jobs in the immediate backward job chain $\overleftarrow{\varsigma^{j,S,\ell}}$ that are executed on the same ECU. There are two scenarios to be considered: (1) $J_{E_j(i+1),q}$ arrives *before* $J_{E_j(i),p}$ finishes in schedule S, and (2) $J_{E_j(i+1),q}$ arrives *not before* $J_{E_j(i),p}$ finishes in schedule S. Since S, ℓ and j are fixed in the statement of the lemma, we dropped these indexes for the simplicity of presentation.

- **Case 1** $a_{i+1} < f_i$, i.e., $J_{E_j(i+1),q}$ arrives before $J_{E_j(i),p}$ finishes. Hence, Eq. (7) holds.
- **Case 2** $a_{i+1} \geq f_i$, i.e., $J_{E_j(i+1),q}$ arrives at or after $J_{E_j(i),p}$ finishes. By our assumption that each task releases its jobs with a bounded maximum inter-arrival time, there must be another job $J_{E_j(i),p+1}$ after $J_{E_j(i),p}$, i.e., $J_{E_j(i),p+1}$ exists. By the definition of the immediate backward job chain, job $J_{E_j(i),p+1}$ cannot finish before $J_{E_j(i+1),q}$ starts its execution; otherwise job $J_{E_j(i),p+1}$ should be in the immediate backward job chain $\overrightarrow{\varsigma^{j,S,\ell}}$ instead of job $J_{E_j(i),p}$ since $J_{E_j(i),p+1}$ finishes later than $J_{E_j(i),p}$ and finishes before $J_{E_j(i+1),q}$ starts its execution. Let r be the arrival time of job $J_{E_j(i),p+1}$ in schedule S. By the above discussion, $r + R_{E_j(i)} \geq \delta_{i+1} \geq a_{i+1}$, i.e., the arrival time of job $J_{E_j(i),p+1}$ plus the worst-case response time of the job must be no less than $\delta_{i+1} \leq a_{i+1}$. Moreover, by the definition of maximum inter-arrival time, we know that $r \leq a_i + T_{E_j(i)}^{max}$. Combining these two inequalities, we get

$$a_{i+1} - a_i = (a_{i+1} - r) + (r - a_i) \leq R_{E_j(i)} + T_{E_j(i)}^{max} \tag{14}$$

  The inequality in Eq. (14) can be improved when $E_j(i) < E_j(i+1)$, i.e., the priority of task $\tau_{E_j(i)}$ is higher than task $\tau_{E_j(i+1)}$. If this holds, the arrival time r of job $J_{E_j(i),p+1}$ must be later than the arrival time of job $J_{E_j(i+1),q}$, i.e., $r > a_{i+1}$; otherwise, the finishing time of $J_{E_j(i),p+1}$ in schedule S must be earlier than δ_i according to the preemptive fixed-priority scheduling strategy, which contradicts to the definition of the immediate backward job chain. Therefore, by $r > a_{i+1}$ and $r \leq a_i + T_{E_j(i)}^{max}$, we have

$$a_{i+1} - a_i < r - a_i \leq T_{E_j(i)}^{max} \text{ if } [E_j(i) < E_j(i+1)] \tag{15}$$

These two cases and the assumption $R_{E_j(i)} \leq T_{E_j(i)}^{max}$ result in

$$\begin{aligned} a_{i+1}^{j,S,\ell} - a_i^{j,S,\ell} &\leq \max\left\{R_{E_j(i)}, T_{E_j(i)}^{max} + R_{E_j(i)} \cdot [E_j(i) > E_j(i+1)]\right\} \\ &\leq T_{E_j(i)}^{max} + R_{E_j(i)} \cdot [E_j(i) > E_j(i+1)] \end{aligned} \tag{16}$$

which concludes the proof. □

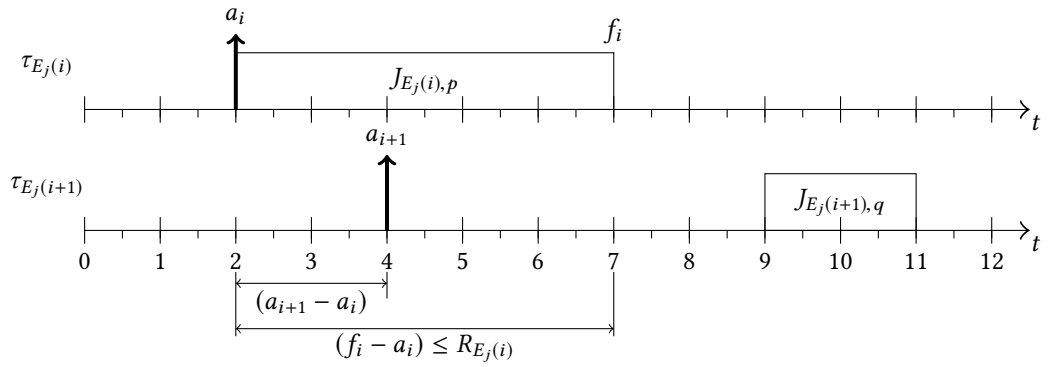
Lemma 5.8. *Suppose that* $J_{E_j(i),p}$ *and* $J_{E_j(i+1),q}$ *are the* i^{th} *and* $(i+1)^{th}$ *jobs in the immediate backward job chain* $\overleftarrow{\varsigma^{j,S,\ell}}$ *that are executed on different ECUs. Then, Eq.* (13) *needs to be extended to:*

$$a_{i+1}^{j,S,\ell} - a_i^{j,S,\ell} \leq T_{E_j(i)}^{max} + R_{E_j(i)} \cdot [P] \tag{17}$$

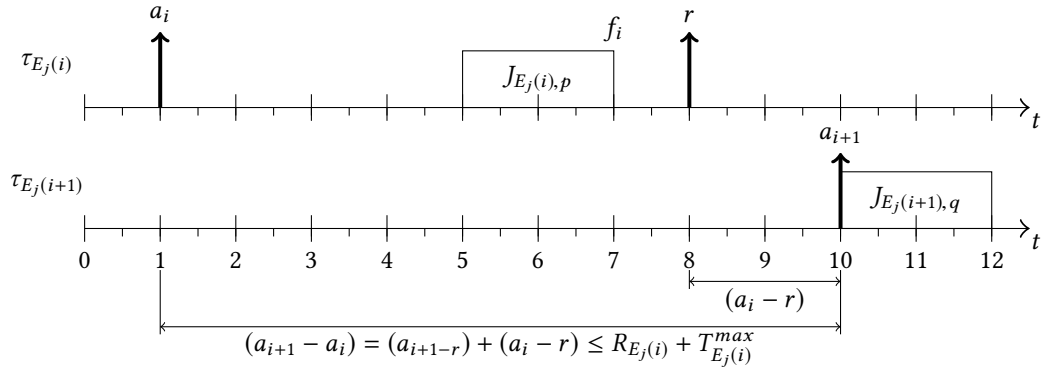
where $[P]$ *is the Iverson bracket, with the following proposition:*

$$[P] = \begin{cases} 1 & \text{if } E_j(i) > E_j(i+1) \vee ECU(E_j(i)) \neq ECU(E_j(i+1)) \\ 0 & \text{otherwise} \end{cases}$$

In other words, the Iverson bracket $[P]$ *is* 1 *when the* $(i+1)^{th}$ *task in the cause-effect chain* E_j *has a higher priority or is executed on a different ECU than the* i^{th} *task in the cause-effect chain,* 0 *otherwise.*



(a) Case 1: $a_{i+1} < f_i$, i.e., $J_{E_j(i+1),q}$ arrives before $J_{E_j(i),p}$ finishes.



(b) Case 2: $a_{i+1} \geq f_i$, i.e., $J_{E_j(i+1),q}$ arrives at or after $J_{E_j(i),p}$ finishes.

Fig. 5. Visualization of the two considered scenarios in the maximum data age time delay analysis.

Proof. This follows from Lemma 5.7, since Eq. (7) and (14) hold. The improvement in Eq. (15) can not be applied and is therefore extended, such that we reach the inequality in Eq. (17). □

The gap between the release time of two jobs in an immediate backward job chain leads to an upper bound of the worst-case length of the immediate backward job chains ($WCBC_j$).

Theorem 5.9. *Let R_k of task τ_k be no more than T^{min}_k for every task τ_k in Γ. For a cause-effect chain E_j,*

$$WCBC_j \leq R_{E(K_j)} + \sum_{i=1}^{K_j - 1} T^{max}_{E_j(i)} + R_{E_j(i)} \cdot [P] \tag{18}$$

Proof. Let $J_{E_j(K_j),\ell^*}$ be the first job of $\tau_{E_j(K_j)}$ that can define a valid immediate backward job chain in a schedule S. By definition

$$\begin{aligned} WCBC_j &= \max_S \max_{\overleftarrow{\varsigma^{j,S,\ell}}, \forall \ell = \ell^*, \ell^*+1, \dots} f_{K_j}^{j,S,\ell} - a_1^{j,S,\ell} \\ &= \max_S \max_{\overleftarrow{\varsigma^{j,S,\ell}}, \forall \ell = \ell^*, \ell^*+1, \dots} f_{K_j}^{j,S,\ell} - a_{K_j}^{j,S,\ell} + \left(\sum_{i=1}^{K_j-1} a_{i+1}^{j,S,\ell} - a_i^{j,S,\ell} \right) \\ &\leq R_{E(K_j)} + \sum_{i=1}^{K_j-1} T_{E_j(i)}^{max} + R_{E_j(i)} \cdot [P] \end{aligned} \tag{19}$$

where the $\leq$ is due to Lemma 5.8. □

Theorem 5.10. *Suppose that R_k of task τ_k is no more than T_k^{min} for every task τ_k in Γ. The maximum data age of a cause-effect chain E_j is upper bounded by the right-hand side of Eq. (18).*

Proof. This is due to Definition 4.1 and Theorem 5.9. □

6 RELATION BETWEEN REACTION TIME AND DATA AGE

In this section, we take a closer look at the relation between maximum reaction time and maximum data age. We show a strict relation between the maximum data age and the maximum response time, i.e., that the maximum reaction time is always an upper bound of the maximum data age. At first it is presented that this relation holds for the bounds we provided in Section 5, and afterwards we consider the general case.

Theorem 6.1. *For a cause-effect chain E_j, the upper bound on the maximum reaction time in Theorem 5.4 is a strict upper bound of the upper bound on the maximum data age in Theorem 5.10.*

Proof. The upper bound on the maximum data age of E_j from Theorem 5.10 is $R_{E(K_j)} + \sum_{i=1}^{K_j-1} \left(T_{E_j(i)}^{max} + R_{E_j(i)} \cdot [P] \right)$. The upper bound on the maximum reaction time of E_j from Theorem 5.4 is

$$\begin{aligned} & T_{E_j(1)}^{max} + R_{E(K_j)} + \sum_{i=1}^{K_j-1} \max \left\{ R_{E_j(i)}, T_{E_j(i+1)}^{max} + R_{E_j(i)} \cdot [P] \right\} \\ \geq\ & T_{E_j(1)}^{max} + R_{E(K_j)} + \sum_{i=1}^{K_j-1} \left(T_{E_j(i+1)}^{max} + R_{E_j(i)} \cdot [P] \right) \\ =\ & R_{E(K_j)} + \sum_{i=1}^{K_j-1} \left(T_{E_j(i)}^{max} + R_{E_j(i)} \cdot [P] \right) + T_{E_j(K_j)}^{max} \\ >\ & R_{E(K_j)} + \sum_{i=1}^{K_j-1} \left(T_{E_j(i)}^{max} + R_{E_j(i)} \cdot [P] \right) \end{aligned} \tag{20}$$

where the $\geq$ is due to the removal of the maximum operator. □

While the proofs for Theorem 5.4 and Theorem 5.10 are straightforward, the general case needs some additional consideration.

Theorem 6.2. *The maximum data age of a cause-effect chain E_j is always (strictly) upper-bounded by its maximum reaction time.*

Proof. Suppose that the maximum data age of a cause-effect chain is observed in an immediate backward job chain $\overleftarrow{\varsigma^{j,S,\ell}}$ that ended at the ℓ-th job of task $\tau_{E_j(K_j)}$ in a schedule S. Suppose that the chain $\overleftarrow{\varsigma^{j,S,\ell}}$ starts from job $J_{E_j(1),p}$, which arrives at time $a_1^{j,S,\ell}$. The maximum data age is hence $f_{K_j}^{j,S,\ell} - a_1^{j,S,\ell}$.

We consider the immediate forward job chain $\overrightarrow{\varsigma^{j,S,p+1}}$ starting from job $J_{E_j(1),p+1}$. We claim that the job of $\tau_{E_j(i)}$ in the immediate forward job chain $\overrightarrow{\varsigma^{j,S,p+1}}$ must arrive after the job of task $\tau_{E_j(i)}$ in the immediate backward job chain $\overleftarrow{\varsigma^{j,S,\ell}}$ for $i = 1, 2, \ldots, K_j$. If this holds, the maximum reaction time is at least $T_{E_j(1)}^{max}$ plus the length of the immediate forward job chain $\overrightarrow{\varsigma^{j,S,p+1}}$, which is strictly more than $f_{K_j}^{j,S,\ell} - a_1^{j,S,\ell}$, i.e., the maximum data age of E_j.

We now prove the above claim by contradiction. Suppose for contradiction that index k is the smallest integer in which the job of $\tau_{E_j(k)}$ in the immediate forward job chain $\overrightarrow{\varsigma^{j,S,p+1}}$ arrives no later than the job of task $\tau_{E_j(k)}$ in the immediate backward job chain $\overleftarrow{\varsigma^{j,S,\ell}}$. Recall the definition of the immediate backward job chain in Section 3. The immediate backward job chain is built based on iteratively searching the jobs of the previous tasks in the chain, which finished as the last job before the job of the next task in the chain starts. Therefore, the construction of $\overleftarrow{\varsigma^{j,S,\ell}}$ for $i = k-1, k-2, \ldots, 1$ enforces that the job of $\tau_{E_j(i)}$ in the immediate backward job chain $\overleftarrow{\varsigma^{j,S,\ell}}$ arrives no earlier than the job of task $\tau_{E_j(i)}$ in the immediate forward job chain $\overrightarrow{\varsigma^{j,S,p+1}}$. That is, the immediate backward job chain defines a job of $\tau_{E_j(1)}$ that arrives no earlier than $J_{E_j(1),p+1}$ (i.e., later than job $J_{E_j(1),p}$) as the first job in the immediate backward job chain. This contradicts to the condition that $J_{E_j(1),p}$ is the first job in the immediate backward job chain $\overleftarrow{\varsigma^{j,S,\ell}}$.

Therefore, the theorem is proved. □

7 EVALUATION

To evaluate the methods derived in Section 5, we compare the resulting values for maximum reaction time and maximum data age with the upper bound by Davare et al. [9]. Note that Eq. (5) is mainly discussed for the maximum reaction time in [9], but it can be also applied to calculate the maximum data age. We report the precision gain, defined as $\frac{bound_{Davare} - bound_{ours}}{bound_{Davare}} \cdot 100$, where $bound_{ours}$ is Theorem 5.4 for the maximum reaction time or Theorem 5.10 for the maximum data age, while $bound_{Davare}$ is the result from Eq. (5).

In Subsection 7.1, we evaluate the precision gain based on the real world automotive benchmark provided by Kramer et al. [18]. This benchmark proposes a method to generate task sets that have realistic application characteristics of real-world automotive software systems. Furthermore, in Subsection 7.2, we evaluate a wider range of embedded real-time systems by evaluating randomized task sets according to the UUnifast method [6].

To display variation in the precision gain, the analyses results are presented by box plots. The median of each box plot is colored in red. The black box represents the interval around the median that contains the inner 50% of the precision gain, while the whiskers display the range of the top/bottom 25% of the improvement. For the maximum reaction time analysis, the scale of the y-axis ranges from 0% to 35%, whereas the y-axis of the maximum data age analysis ranges from 0% to 80%.

Period	Share	ACET in μs			WCET factor	
		Min	Avg.	Max	f_{min}	f_{max}
1 ms	3%	0.34	5.00	30.11	1.30	29.11
2 ms	2%	0.32	4.20	40.69	1.54	19.04
5 ms	2%	0.36	11.04	83.38	1.13	18.44
10 ms	25%	0.21	10.09	309.87	1.06	30.03
20 ms	25%	0.25	8.74	291.42	1.06	15.61
50 ms	3%	0.29	17.56	92.98	1.13	7.76
100 ms	20%	0.21	10.53	420.43	1.02	8.88
200 ms	1%	0.22	2.56	21.95	1.03	4.90
1000 ms	4%	0.37	0.43	0.46	1.84	4.75

Table 3. The information to generate the automotive task sets, combined from Table III, Table IV, and Table V in "Real World Automotive Benchmarks For Free" [18].

7.1 Synthesized Automotive Systems

Since End-To-End latency analyses are highly important for the overall timing verification in the automotive domain, we evaluated our proposed analyses using synthesized automotive task sets according to the details in "Real World Automotive Benchmarks For Free", provided by Kramer et al. [18] in 2015. The relevant parameters for the task set generation are summarized in Table 3.

Each task was generated in the following manner: the period of a task was drawn from $A = \{1, 2, 5, 10, 20, 50, 100, 200, 1000\}$ according to the related probability distributions.[3] The average execution time (ACET) of a task was based on a Weibull distribution that fulfills the properties as given in Table 3. Afterwards, the ACETs were multiplied with a value drawn equally distributed from the range of the WCET factors for the related period. The utilization of each task τ_i was given by C_i/T_i^{min}, thus the utilization of n tasks is: $\sum_{i=1}^{n} C_i/T_i^{min}$. We generated the task sets with different target utilizations (50%, 60%, 70%, 80%, and 90%), therefore we used a subset-sum approximation algorithm. Namely, we seeded an initial task set with 3000 tasks and calculated a subset of these initial tasks, such that its cumulative utilization was within a specified threshold of the targeted utilization, i.e., 0.1%. Further details about this generation process and reasons why this approach was chosen can be found in [29]. In total, 1000 task sets for each target utilization were generated. The cardinality of the task set was on average ≈ 75 tasks. The cause-effect chains were determined according to Section IV-E in [18]. To be more precise, for each task set Γ_i we created a set of cause-effect chains Π_i that included E_j cause-effect chains, with $j = 1, 2, \cdots, n$, where n is a uniform distributed number between 30 and 60. The specific tasks of E_j were selected by the following steps:

(1) The number of involved activation patterns (1–3) P_j was specified according to the distribution shown in Table VI in [18].
(2) K_j was specified following Table VII in [18], which lists the distribution of the tasks per activation pattern $(2-5)$.
(3) A uniform distributed random choice selected the specific activation patterns in size of P_j from A without replacement.

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

(4) A uniform distributed random choice selected task of each specific period was added to E_j as long as the size of $E_j \neq K_j$, i.e., all specific selected tasks had the same share among the specific periods.

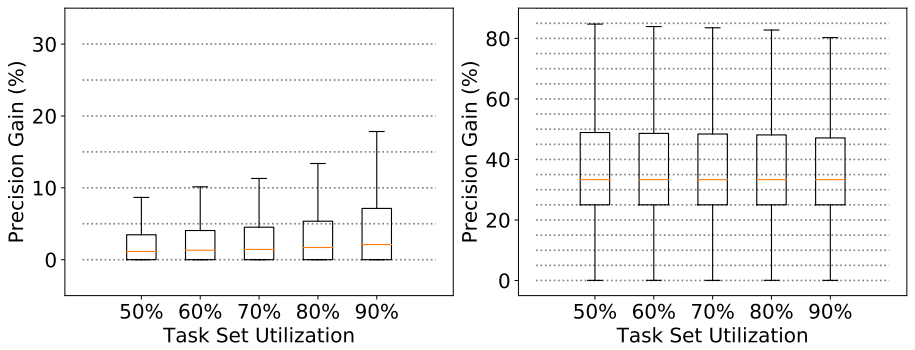
Figure 6 depicts the analyses results for each target utilization. We separate the results into maximum reaction time (6a) and maximum data age (6b). Since the task set generation due to [18] is dependent on a big variety of distributions, the variance of the precision gain for both End-to-End semantics is wide. The bottom whisker shows a minimal gain of 0% for all simulations, which happens if the Iverson bracket in Theorem 5.4 or Theorem 5.10 is equal to 1. For the maximum reaction time the precision gain increases up to $\approx$ 17%, whereas the median gain remains at $\approx$ 2% for all target utilizations. The increase is larger for higher utilizations, because a larger system utilization induces larger worst-case response times of the tasks that can be dropped, if the Iverson bracket is equal to 0. Hence, the precision gain of our analysis becomes more significant. The analysis of the maximum data age shows a larger variance in the possible precision gain as the maximum reaction time, which is in between 0% to 80%, with a constant median of $\approx$ 34%.

7.2 UUnifast Task Set Setup

We conducted evaluations based on four task set parameters. We analyzed the effect of them individually by fixing three of the parameters in each simulation:

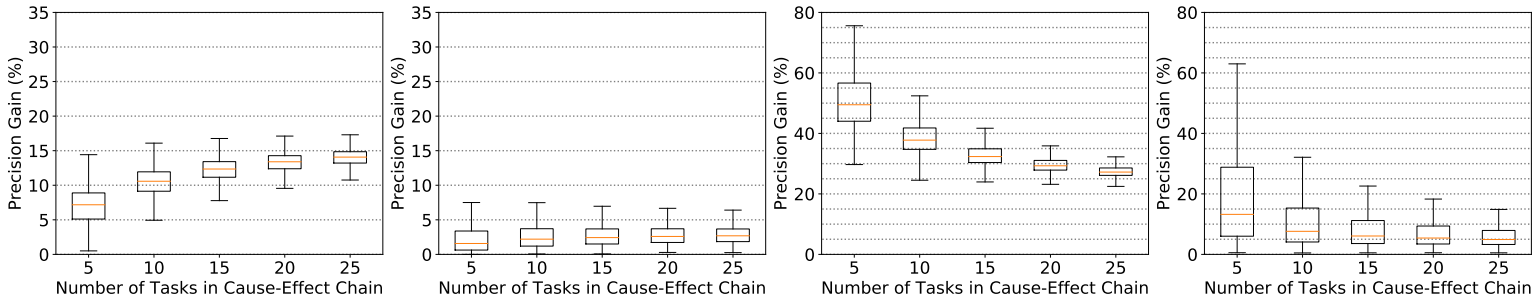
- U: The utilization of the task set on one ECU, given by $\sum_{i=1}^{n} C_i/T_i^{min}$ for $n = 1000$ tasks. When fixed, $U = 70\%$.
- ra: The minimum inter-arrival time T_i^{min} was determined according to a log-uniform distribution over $[1, ra]$. When fixed, $ra = 100$
- K_j: The number of tasks in cause-effect chain E_j. When fixed, $K_j = 25$ if $m = 0$ and $5 \cdot (m+1)$ when the number of communications was evaluated, i.e., Figure 10.
- m: The number of communications between different ECUs. When fixed, $m = 0$, i.e., the chain was executed on one ECU with no communication.

We generated 100 synthesized implicit deadline task sets, i.e., $D_i = T_i$ for all tasks, each corresponding to one ECU, based on the above four parameters. Each task set consisted of 1000 individual tasks, which represented the size of a common industrial application, as suggested in [18]. To be more precise, the utilization values for each task were determined based on the UUniFast algorithm [6] under a given utilization for each individual setup. Afterwards, for each task, T_i^{min} was randomly drawn from the interval $[1, ra]$ based on a log-uniform distribution, as suggested



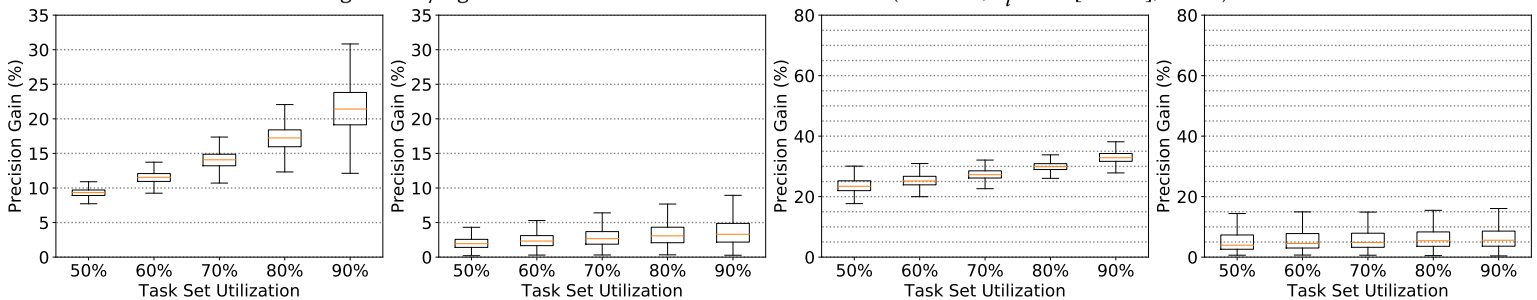
(a) reaction time (b) data age

Fig. 6. Analyses result, considering real-world automotive task set properties



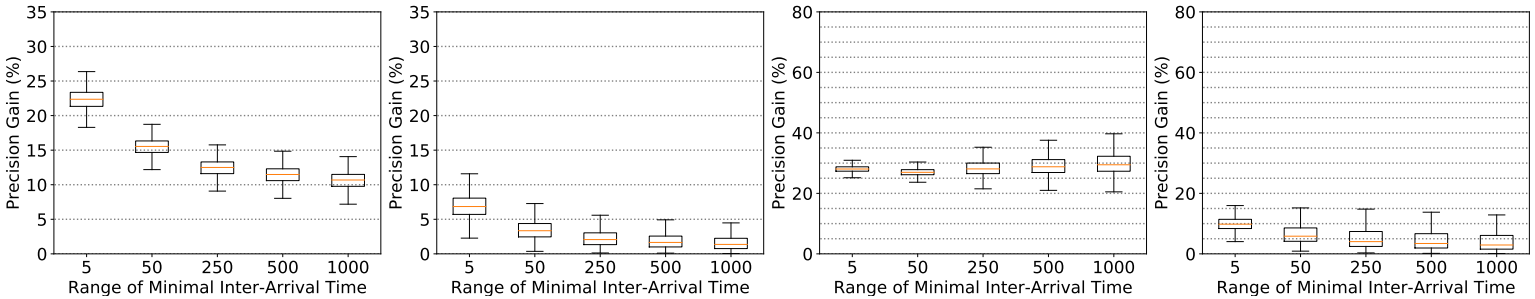
(a) reaction time, priority-ordered (b) reaction time, arbitrarily-ordered (c) data age, priority-ordered (d) data age, arbitrarily-ordered

Fig. 7. Varying the number of tasks in cause-effect chain. ($U = 70\%$, $T_i^{min} = [1 : 100]$, $m = 0$)



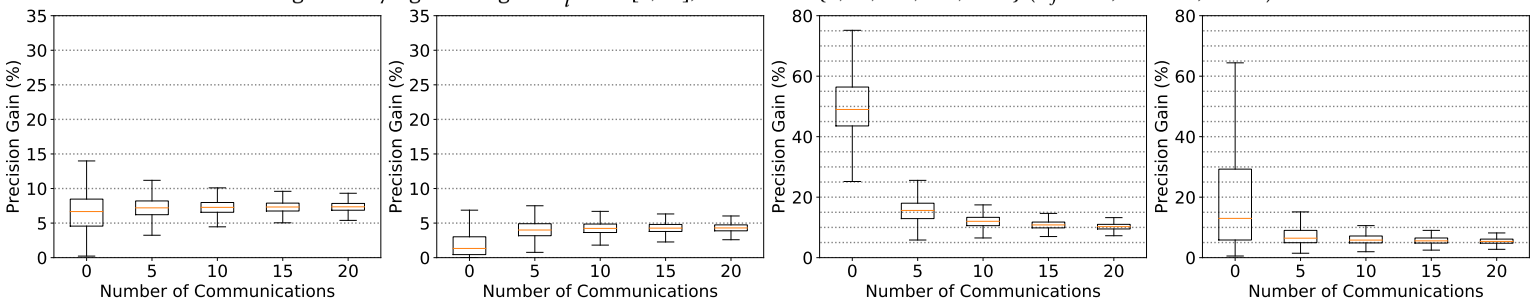
(a) reaction time, priority-ordered (b) reaction time, arbitrarily-ordered (c) data age, priority-ordered (d) data age, arbitrarily-ordered

Fig. 8. Varying the task set utilization. ($K_j = 25$, $T_i^{min} = [1 : 100]$, $m = 0$)



(a) reaction time, priority-ordered (b) reaction time, arbitrarily-ordered (c) data age, priority-ordered (d) data age, arbitrarily-ordered

Fig. 9. Varying the range of $T_i^{min} \in [1, ra]$, where $ra \in \{5, 50, 250, 500, 1000\}$ ($K_j = 25$, $U = 0.7$, $m = 0$)



(a) reaction time, priority-ordered (b) reaction time, arbitrarily-ordered (c) data age, priority-ordered (d) data age, arbitrarily-ordered

Fig. 10. Varying the number of communications between different ECUs. ($K_j = 5 \cdot (m + 1)$, $U = 70\%$, $T_i^{min} = [1 : 100]$)

by Emberson et al. [10]. Then, C_i was set to $U_i \cdot T_i$. To determine T_i^{max}, we drew a uniformly distributed value x_i from $[1, 2]$ for each task and set $T_i^{max} = x_i T_i^{min}$. On each ECU, we considered rate-monotonic scheduling, i.e., tasks with smaller period have higher priority where ties were broken arbitrarily, and determined the WCRT of each task according to the TDA in Eq. (1), detailed in Section 2.1.

Two types of cause-effect chains were considered:

- **priority-ordered chains**: the priority among all tasks in a cause-effect chain is decreasing, where the first task in the cause-effect chain has the highest priority, i.e., the priority of $\tau_{E_j(i)}$ is higher than $\tau_{E_j(i+1)}$ for $i = 1, \cdots, K_j$, if $\tau_{E_j(i)}$ and $\tau_{E_j(i+1)}$ are on the same ECU.
- **arbitrarily-ordered chains**: otherwise.

For each generated task set, we created 200 cause-effect chains of the size K_j by drawing different tasks under a uniform distribution. If priority-ordered chains were considered in the evaluation, the 200 chains were all priority-ordered; otherwise, the 200 chains were all arbitrarily-ordered.

When m communications between cause-effect chains of size K_j located on different ECUs were considered, we generated an independent set of m communication tasks as how we generated task sets previously. For each communication task, a message was generated and inserted between the communicating cause-effect chains. The inter-arrival time of the message was equal to the inter-arrival time of its predecessor in the chain. Furthermore, the WCRT of the message was set, such that it was less than its inter-arrival time.

In **Figure 7**, we consider $U = 70\%$, $T_i^{min} = [1 : 100]$, $m = 0$ by varying $K_j \in \{5, 10, 15, 20, 25\}$. The impact of an increasing number of tasks in a cause-effect chain enlarges the improvement of the maximum reaction time analysis with priority-ordered chains in the range of $\approx 7.5\%$ to $\approx 14\%$. Our analysis allows to exclude the WCRT from the calculated value for all tasks in the chain except the last task. Hence, it can be excluded for 4 out of $K_j = 5$ tasks, i.e., 80% of the tasks, and for 24 out of $K_j = 25$ tasks, i.e., 96%. This means that the effect becomes more significant when K_j increases. However, for the maximum data age, longer chains reduce the precision gain. The precision gain of the maximum data age decreases with an increasing number of tasks in a cause-effect chain from $\approx 50\%$ to $\approx 38\%$ for priority-ordered chains and from $\approx 12\%$ to $\approx 5\%$ for arbitrarily-ordered chains. This is mainly because the excluded maximum inter-arrival time becomes relatively small compared to the calculated data age value if the number of tasks in chain becomes higher.

In **Figure 8**, we consider $K_j = 25$, $T_i^{min} = [1 : 100]$, $m = 0$, by varying $U \in \{50, 60, 70, 80, 90\}$. Since the larger task utilization implies in general larger worst-case response times of the tasks, the precision gain of our analysis becomes more significant when the utilization increases. This observation holds for the maximum reaction time and the maximum data age.

In **Figure 9**, we consider $K_j = 25$, $U = 70\%$, $m = 0$, by setting to $T_i^{min} \in [1, x]$, where $x \in \{5, 50, 250, 500, 1000\}$. The precision gain is better when x is smaller. When x is larger, we can imagine that R_i / T_i^{min} is somehow smaller. This explains the general trend of the decreasing precision gain when x increases.

In **Figure 10**, we consider $K_j = 5$, $U = 70\%$, $T_i^{min} = [1 : 100]$, by varying $m \in \{0, 5, 10, 15, 20\}$. In such a case, the number of tasks in a cause-effect chain increases when m increases. However, since we analyzed a set of individual chains, contrary to the case displayed in Figure 7, there is no significant gain for the maximum reaction time.

Overall, the improvement of the ordered cause-effect chains is higher than that of the arbitrary-ordered chains. This is due to condition P in the Iverson bracket in Theorems 5.4 and 5.10. Furthermore, the improvement of the maximum data age is higher due to the discussions in Section 6.

8 CONCLUSION

In this paper, we provide analyses of the maximum reaction time and the maximum data age for the data propagation of a cause-effect chain in a distributed system. Our analyses are based on immediate forward and backward job chains, which are in our opinion more natural than the data freshness perspectives in the literature. Our analytical results dominate the state of the art by Davare et al. [9], and significant improvements can be observed in our evaluations.

ACKNOWLEDGMENTS

This paper is supported by DFG, as part of the Collaborative Research Center SFB876, project A1 (http://sfb876.tu-dortmund.de/).

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 *7th International 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 *22nd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA 2016, Daegu, South Korea, August 17-19, 2016*, 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 (invited paper). In *IEEE 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 *Embedded Software, Second International Conference, 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] A. Burns. Preemptive priority-based scheduling: An appropriate engineering approach. In *Advances in Real-Time Systems, chapter 10*, pages 225–248. Prentice Hall, 1994.

[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] P. Emberson, R. Stafford, and R. I. Davis. Techniques for the synthesis of multiprocessor tasksets. In *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010)*, pages 6–11, 2010.

[11] 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.

[12] J. Forget, F. Boniol, and C. Pagetti. Verifying end-to-end real-time constraints on multi-periodic models. In *22nd IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, pages 1–8, 2017.

[13] 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.

[14] A. Hamann, D. Dasari, S. Kramer, M. Pressler, and F. Wurst. Communication centric design in complex automotive embedded systems. In *Euromicro Conference on Real-Time Systems, ECRTS*, pages 10:1–10:20, 2017.

[15] T. Klaus, F. Franzmann, M. Becker, and P. Ulbrich. Data propagation delay constraints in multi-rate systems: Deadlines vs. job-level dependencies. In *Proceedings of the 26th International Conference on Real-Time Networks and Systems*, pages 93–103. ACM, 2018.

[16] T. Kloda, A. Bertout, and Y. Sorel. Latency analysis for data chains of real-time periodic tasks. In *23rd IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, pages 360–367, 2018.

[17] H. Kopetz. *Real-Time Systems - Design Principles for Distributed Embedded Applications.* Real-Time Systems Series. Springer, 2011.

[18] S. Kramer, D. Ziegenbein, and A. Hamann. Real world automotive benchmark for free. In *6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2015.

[19] J. P. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *IEEE Real-Time Systems Symposium'89*, pages 166–171, 1989.

[20] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[21] A. K. Mok. Fundamental design problems of distributed systems for the hard-real-time environment. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1983.

[22] S. Mubeen, J. Mäki-Turja, and M. Sjödin. Implementation of end-to-end latency analysis for component-based multi-rate real-time systems in rubus-ice. In *Factory Communication Systems (WFCS), 2012 9th IEEE International Workshop on*, pages 165–168. IEEE, 2012.

[23] S. Mubeen, J. Mäki-Turja, and M. Sjödin. Translating end-to-end timing requirements to timing analysis model in component-based distributed real-time systems. *SIGBED Review*, 9(4):17–20, 2012.

[24] 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 *Proceedings of the tenth ACM international conference on Embedded software*, pages 129–138. ACM, 2010.

[25] 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.

[26] Symtavision. SymTA/S Toolbox. http://www.symtavision.com/symtas.html.

[27] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming*, 40(2-3):117–134, 1994.

[28] G. von der Bruggen, J.-J. Chen, and W.-H. Huang. Schedulability and optimization analysis for non-preemptive static priority scheduling based on task utilization and blocking factors. In *Euromicro Conference on Real-Time Systems, ECRTS*, pages 90–101, 2015.

[29] G. von der Brüggen, N. Ueter, J. Chen, and M. Freier. Parametric utilization bounds for implicit-deadline periodic tasks in automotive systems. In *Proceedings of the 25th International Conference on Real-Time Networks and Systems, RTNS*, pages 108–117, 2017.