End-To-End Latency of Cause-Effect Chains: A Tutorial

MARIO GÜNZEL, Faculty of Informatics, TU Dortmund University, Dortmund, Germany
HARUN TEPER, Faculty of Informatics, TU Dortmund, Dortmund, Germany
GEORG VON DER BRÜGGEN, Faculty of Informatics, TU Dortmund University, Dortmund, Germany
JIAN-JIA CHEN, Faculty of Informatics, TU Dortmund University, Dortmund, Germany

In many applications of cyber-physical systems, a sequence of tasks is necessary to perform a certain functionality. For example, from a sensor to an actuator, the first task reads the sensor value (cause), the second task processes the data, and the third task produces an output for the actuator (an effect is triggered). For such scenarios, the *end-to-end* timing properties (the so-called end-to-end latency) of the sequence of tasks (the so-called cause-effect chain) are of importance. This tutorial recaps different metrics for the end-to-end latency of cause-effect chains, and summarizes fundamental properties and existing analytical results in a systematic manner. To that end, this tutorial has a special focus on the reaction time (how fast can a reaction be in the worst case) and the data age (how old is the data source of an actuation in the worst case). The goal of this tutorial is to provide a systematic view of the fundamental end-to-end timing properties of cause-effect chains and offer an outlook of possible research directions in the near future. Furthermore, we extend the proof of one fundamental property in the literature to comply with the current state-of-the-art definition of end-to-end latencies.

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

Additional Key Words and Phrases: End-to-end, cause-effect chain, distributed systems, maximum data age, maximum reaction time, explicit communication, implicit communication, logical execution time

ACM Reference Format:
Mario Günzel, Harun Teper, Georg von der Brüggen, and Jian-Jia Chen. 2024. End-To-End Latency of Cause-Effect Chains: A Tutorial. *ACM Trans. Embedd. Comput. Syst.* 24, 1, Article 22 (December 2024), 18 pages. https://doi.org/10.1145/3703630

This work was part of a project (PropRT) that received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 865170). The work was partially funded by the German Federal Ministry of Education and Research (BMBF) in the course of the 6GEM research hub under grant number 16KISK038. The work was also supported by Deutsche Forschungsgemeinschaft (DFG), as part of the Collaborative Research Center SFB876, project A1 (http://sfb876.tu-dortmund.de/) and Sus-Aware (project no. 398602212).
Authors' Contact Information: Mario Günzel, Faculty of Informatics, TU Dortmund University, Dortmund, Nordrhein-Westfalen, Germany; e-mail: mario.guenzel@tu-dortmund.de; Harun Teper, Faculty of Informatics, TU Dortmund, Dortmund, Nordrhein-Westfalen, Germany; e-mail: harun.teper@tu-dortmund.de; Georg von der Brüggen, Faculty of Informatics, TU Dortmund University, Dortmund, Nordrhein-Westfalen, Germany; e-mail: georg.von-der-brueggen@tu-dortmund.de; Jian-Jia Chen, Faculty of Informatics, TU Dortmund University, Dortmund, Nordrhein-Westfalen, Germany; e-mail: jian-jia.chen@cs.uni-dortmund.de.

ACM 1539-9087/2024/12-ART22
https://doi.org/10.1145/3703630

1 Introduction

Over the past few decades, computation has become ubiquitous. Multiple obvious computation systems—like laptops, smartphones, tablets, and smart TVs—are part of and play an important role in our everyday life. However, short intervals where such systems do not work as intended usually only reduce the quality of the provided service but do not endanger the device, the device users, other people, or the environment. Therefore, it is normally sufficient to provide a best-effort approach when ensuring the system's functionality.

However, computation systems have a crucial yet often less obvious role in our everyday life. Today, they are deeply embedded into and (at least partly) control most physical plants and processes. Such systems are called *cyber-physical systems*, as they are built from and depend upon the synergy of computational and physical components. Application areas include smart homes, medical technology, traffic management, airplanes, cars, logistics, and manufacturing, among others.

Most cyber-physical control systems must react to external activities with an appropriate actuation. They are often safety-critical, since not behaving as expected may lead to dire consequences for the system, humans, or the environment. Another characteristic such control systems often have in common is that they perform the same functionality over and over again—frequently checking for an external activity and providing an actuation when needed. Figure 1 depicts an automatic brake system where the car should automatically be stopped when a pedestrian is observed in front of the car. In this simplified example, three steps (each handled by a related task) must be performed: (i) a camera image is taken, (ii) image recognition marks pedestrians in the picture, and (iii) danger analysis determines whether the car must be stopped to not endanger pedestrians. This example also shows why safety-critical control systems often have timing requirements: It is not sufficient to calculate the correct result based on the observed situation (i.e., braking)—the required actuation must also be performed within a certain time interval to not harm the pedestrian.

A system that requires both functional and timing correctness is called a *real-time system*. Achieving timing correctness, on a high level, is a two-step process. First, the recurrent tasks must be assigned to processing units, and if multiple tasks are assigned to the same component, it must be specified how task instances are scheduled to be processed. Second, given a task-to-resource assignment and scheduling policies for the individual resources, it must be determined whether all timing constraints are always fulfilled. Therefore, finding good assignment strategies and scheduling policies as well as providing tests that, for a given resource assignment and scheduling strategy, determine whether all timing requirements are met are both important research directions.

While classical analyses of the *worst-case response time* or the *schedulability* of real-time systems only provide timing guarantees for individual tasks, in many scenarios the interplay between tasks plays an important role as well. Specifically, if a functionality is achieved by several cooperating tasks, transferring produced/required data between the tasks is required, as shown in the example in Figure 1. For such scenarios, we are interested in the timing behavior of the data propagation along a chain that is composed of multiple tasks that must communicate the necessary data to their successor in the chain. We call a related analysis that considers a chain of communicating tasks and determines the time needed to process data through the complete chain an *end-to-end analysis*. Specifically, end-to-end analysis determines the the maximum time for the dataflow between the *cause* (external activity) and the *effect* (actuation based on the external activity). Typical metrics to be analyzed include the following:

— *Maximum reaction time*: **Maximum Reaction Time (MRT)** is the longest time interval from an external cause until the earliest time where this external cause is fully processed (also called the *maximum button-to-action delay*).

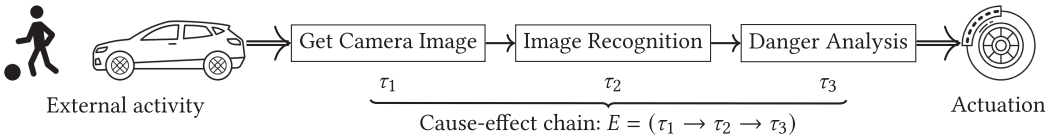

Fig. 1. Simplified automatic brake system.

— *Maximum data age*: The **Maximum Data Age (MDA)** is the longest time interval starting from sampling a value and ending at an effect that is caused by an actuation that processes the sampled data (also called the *data freshness*).

The challenges that end-to-end analysis has to deal with are inherently different from classical response time analysis. Whereas response time analysis focuses on the timing behavior of *individual tasks*, for end-to-end analysis the focus is on the latency of the data propagation along several tasks. While some analytical results for the end-to-end latency may exploit bounds on the response time of tasks, end-to-end analysis additionally has to deal with over- and undersampling. In other words, data may be overwritten (oversampling) or utilized by several jobs (undersampling).

The seminal work by Davare et al. [10] explored the end-to-end latency and optimization of *cause-effect chains* in 2007. Feiertag et al. [14] iterated upon the work of Davare et al. [10] in 2009. AUTOSAR Timing Extensions [2] represent such cause-effect chains of more general functional dependency. Such communication-centric designs are also reflected by Hamann et al. [22] from Robert Bosch, whereas we [41] provide a synergy in **Robot Operating System 2 (ROS2)**.

In the past two decades, extensive results to analyze the end-to-end latency of cause-effect chains have been provided. The specification of the MDA and the MRT has evolved over the years to consider different system models. Furthermore, analytical results under different communication and task release policies have been developed. This tutorial reviews the existing results in this area and (after the short introductions of real-time systems in Section 2 and cause-effect chains in Section 3) serves to

— recap different types and models of end-to-end latency in Section 4,
— provide fundamental timing properties in Section 5,
— summarize existing analytical results in a systematic manner in Section 6, and
— offer an outlook of possible research directions in the near future in Section 7.

To that end, we extend the proof of one fundamental property in Section 5.1 to comply with the current state-of-the-art definition of end-to-end latencies in Section 4.

2 Real-Time Systems

In this section, we provide the basic concepts and definitions regarding task systems and their scheduling which are subsequently utilized when describing and analyzing cause-effect chains.

Task Model. We first introduce the general real-time systems task model and the related notation. We consider a system $\mathbb{T}$ comprised of multiple tasks. Each individual *task* τ is a recurrently activated program that produces output data based on its inputs. Task τ releases (countably many) task instances (called *jobs*), denoted as $(\tau(m))_{m \in \mathbb{N}^+}$ where $\tau(i)$ is the i-th job of τ. Sometimes, for example, if the task of a job is not further specified, we also denote a job as J.

The definitions and some analytical results presented in this tutorial do not rely on any assumptions about the job releases. However, in practical situations, the arrival pattern is often restricted regarding the first time a job of a task arrives or on the inter-arrival pattern of jobs. Hence, the most common task models, namely the periodic and the sporadic task model, impose additional restrictions.

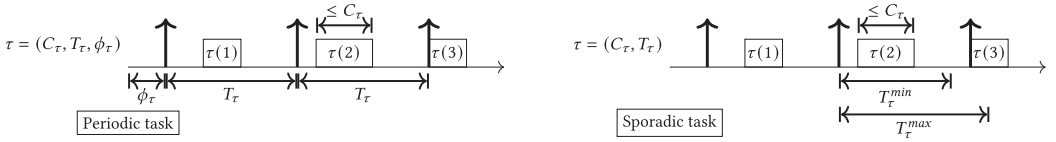

Fig. 2. Task model. Left: Periodic task. Right: Sporadic task.

A *periodic task* τ is specified as $\tau = (C_\tau, T_\tau, \phi_\tau) \in \mathbb{R}^3$, where $C_\tau \geq 0$ is the worst-case execution time, $T_\tau > 0$ is the period, and ϕ_τ is the phase. The task τ releases its first job $\tau(1)$ at time $r_{\tau(1)} = \phi_\tau$ and subsequent jobs every T_τ time units—that is, $\tau(m)$ is released at time $r_{\tau(m)} = \phi_\tau + (m-1) \cdot T_\tau$. Every job of τ executes for at most C_τ time units. A periodic task τ is depicted on the left-hand side of Figure 2.

A *sporadic task* τ is specified as $\tau = (C_\tau, T_\tau^{min}) \in \mathbb{R}^2$, where $C_\tau \geq 0$ is the worst-case execution time and $T_\tau^{min} > 0$ is the minimum inter-arrival time between two jobs. Specifically, if job $\tau(i)$ is released at time $r_{\tau(i)}$, then $\tau(i+1)$ is released no earlier than at time $r_{\tau(i)} + T_\tau^{min}$. Each job executes for at most C_τ time units. For end-to-end analysis with sporadic tasks, we further require that a task τ also has a specified maximum inter-arrival time $T_\tau^{max} \geq T_\tau^{min}$. Specifically, we write $\tau = (C_\tau, T_\tau^{min}, T_\tau^{max})$ in that case. Without a maximum inter-arrival time, after a certain point in time no more jobs of τ may be released and the end-to-end timing would be infinite—which should be prevented by design. Such an extended sporadic task is depicted on the right-hand side of Figure 2.

Usually, individual tasks need to adhere to certain timing requirements. In other words, a task τ can be equipped with a *relative deadline* $D_\tau > 0$. Each job of task τ then has to comply with the timing constraint imposed by the relative deadline. More specifically, if job $\tau(i)$ is released at time $r_{\tau(i)}$, then is has to finish until its *absolute deadline* $r_{\tau(i)} + D_\tau$.

Scheduling Model. When multiple tasks are executed on the same processor, it must be decided which job is executed at which point in time. A schedule is non-preemptive if a job always runs to completion before it releases the processor and preemptive if a job can preempt its execution before completion. Static schedulers create a schedule table that allocates time slots to the tasks offline. At runtime, task instances are allocated to the processor based on these time slots and the table is repeated. Alternatively, priority-based schedulers allocate jobs to the processor based on (usually unique) priorities. Here, at any point in time when a scheduling decision is made, the job with the currently highest priority is allocated to the processor.

Priority-based schedulers can be classified into (task-level) fixed-priority and (task-level) dynamic-priority schedulers. In other words, a scheduler is considered a fixed-priority scheduler if, for any two tasks τ_i and τ_j, either all jobs of τ_i have higher priority than all jobs of τ_j or all jobs of τ_j have higher priority all jobs of τ_i. Well-known fixed-priority scheduling policies are rate monotonic, where the task with the smaller period (for periodic tasks) or minimum inter-arrival time (for sporadic tasks) has higher priority, and deadline monotonic, where the task with the shorter relative deadline has higher priority. Alternatively, for dynamic-priority scheduling, a job of τ_i may have a higher priority than a job of τ_j while a job of τ_j may also have a higher priority than a job of τ_i. The most well-known dynamic-priority scheduling policy is earliest-deadline-first scheduling, where job priority is assigned according to the absolute deadline of jobs.

Under a given scheduling mechanism, the worst-case response time R_τ of a task τ can be specified. More specifically, R_τ is the maximum time between the release and finish of any job of τ. A worst-case response time analysis [26, 33] aims to determine (or upper bound) the worst-case response time of a task.

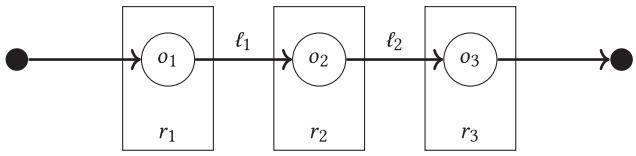

Fig. 3. Dataflow semantic redrawn from the illustration in the work of Davare et al. [10] for completeness.

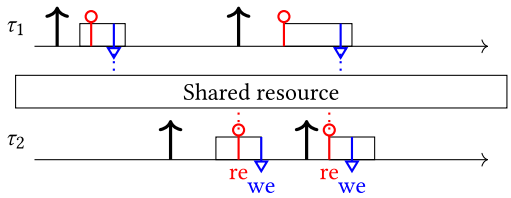

Fig. 4. Communication between tasks.

Note that two of the mentioned scheduling strategies, namely deadline monotonic and earliest-deadline-first scheduling, consider deadlines for the scheduling decision, whereas we do not require tasks to have a deadline in general for cause-effect chains. The reason is that such assumptions and restrictions are unnecessary, as deadlines are not relevant for the end-to-end analysis itself—it is just necessary that the scheduling behavior for the individual processors is known, which may be based on the deadline parameter of the task.

3 Cause-Effect Chains

In general, a *cause-effect chain* is a sequence of objects that need to communicate to achieve a given functionality. One typical example (an automatic brake system that can detect pedestrians on the street) can be found in Figure 1. Cause-effect chains are used to specify a dataflow within a system to be analyzed.

One of the first models to describe dataflow and the end-to-end timing behavior was provided by Davare et al. [10] in 2007. In their model, depicted in Figure 3, objects $O = \{o_i\}$ are allocated to resources $\mathcal{R} = \{r_i\}$. A set of links $\mathcal{L} = \{\ell_i\}$ that each connect two objects describe the *dataflow* between objects. A *path* is a finite sequence of objects with links between them. Today, the dataflow is modeled by a cause-effect chain as a sequence of tasks $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$. Beyond that, modern modeling approaches describe the underlying communication mechanisms more precisely. In this section, we formally introduce cause-effect chains. To that end, we first specify the communication semantics.

Communication. Jobs communicate by receiving (reading) their input data from a shared resource and handing over (writing) their output data to a shared resource. We denote the time of the *read-event* of a job J as $\mathrm{re}(J) \in \mathbb{R}$ and the time of the *write-event* of J as $\mathrm{we}(J) \in \mathbb{R}$. The general communication semantics are depicted in Figure 4. We consider communication under the following two restrictions:

— *(R1) Proper ordering*: For each task $\tau \in \mathbb{T}$, each job $\tau(m)$ reads before it writes—that is, $\mathrm{re}(\tau(m)) \leq \mathrm{we}(\tau(m))$ for all $m \in \mathbb{N}_+$. Moreover, for any two consecutive jobs, the read- and write-events are ordered according to their index—that is, $\mathrm{re}(\tau(m)) < \mathrm{re}(\tau(m+1))$ and $\mathrm{we}(\tau(m)) < \mathrm{we}(\tau(m+1))$ for all $\tau \in \mathbb{T}$ and $m \in \mathbb{N}_+$.

— *(R2) Proper distribution*: The number of read- and write-events in each bounded time interval is finite. More specifically, the sets $\{\mathrm{re}(\tau(m)) \mid m \in \mathbb{N}_+\}$ and $\{\mathrm{we}(\tau(m)) \mid m \in \mathbb{N}_+\}$ have no accumulation point.

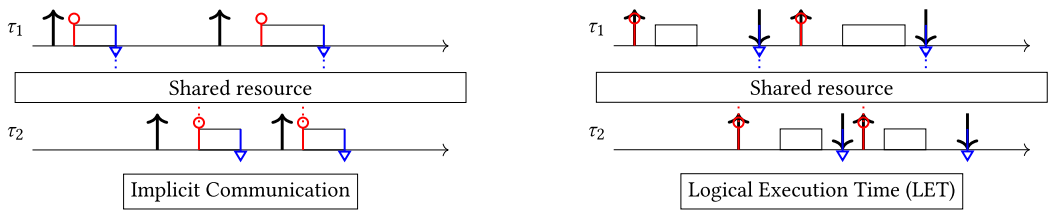

Fig. 5. Implicit communication (left) and LET communication (right).

These two not very restrictive requirements are fulfilled by commonly considered communication semantics—for instance, for *explicit communication*, *implicit communication*, and **Logical Execution Time (LET)**, specified in AUTOSAR [2].

Explicit Communication. Under explicit communication, the read-event and the write-event can occur anywhere during job execution. However, the job cannot write new data before its read-event. An example is depicted in Figure 4.

Implicit Communication. Under implicit communication, each job's read-event is when the job starts (i.e., at the first time the job is executed) and the job's write-event when the job finishes (i.e., at the last time it is executed). Implicit communication is shown on the left-hand side of Figure 5. In contrast to explicit communication, implicit communication strategies easily guarantee data consistency [22].

Logical Execution Time. LET [27] enables deterministic behavior, by performing the jobs' read- and write-events at task-specific offsets. To ensure that read-events occur before the job starts and that write-events occur after the job finishes, a common strategy is to assign each task τ an arbitrary relative deadline D_τ, and to set the read-event of each job J of τ to its release time r_J and the write-event to its absolute deadline $r_J + D_\tau$. Communication under LET is depicted on the right-hand side of Figure 5 with downward arrows indicating absolute deadlines.

To deploy communication functionalities in embedded systems, respecting the introduced communication semantics, programming languages like Giotto [23] or nesC [15] can be utilized.

Cause-Effect Chains. A *cause-effect chain* $E = (\tau_1 \rightarrow \tau_2 \rightarrow \cdots \rightarrow \tau_n)$, with $n \in \mathbb{N}$, describes a data path through different programs as a finite sequence of tasks $\tau_i \in \mathbb{T}$. This definition is inspired by *event chains* specified in the AUTOSAR Timing Extensions [2], which represent chains of more general functional dependency.

We assume that the sampling for a cause-effect chain E happens at the read-event of each job of τ_1 (i.e., implicit sampling) for convenience. Alternatively, any kind of sampling can be modeled by adding a *sampling task* τ^{sample}, where each job $\tau^{sample}(1), \tau^{sample}(2), \ldots$ reads and writes data at a time when the sampling happens. We focus on one specific cause-effect chain in this tutorial. In most systems, the data dependencies are more complex and can be described by a directed acyclic graph with several sources and sinks. In this case, each cause-effect chain (i.e., path through the directed acyclic graph from a source to a sink) can be analyzed individually.

To conduct end-to-end analysis, it is not sufficient to know which tasks are part of the chain. Instead, we need to construct specific data paths of single data tokens through the task instances. We describe these specific data paths as *job chains*.

Definition 3.1 (Job chain). For a cause-effect chain $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$ of the task set $\mathbb{T}$, a job chain $c = (J_1, \ldots, J_n)$ for E is a sequence of jobs fulfilling the following two conditions:

- Job J_i is a job of task τ_i for all $i \in \{1, 2, \ldots, n\}$.
- Each job in the chain reads the data at the time or after it was written by the previous job in the chain. In other words, $\text{we}(J_{i-1}) \leq \text{re}(J_i)$ for all $i \in \{2, 3, \ldots, n\}$.

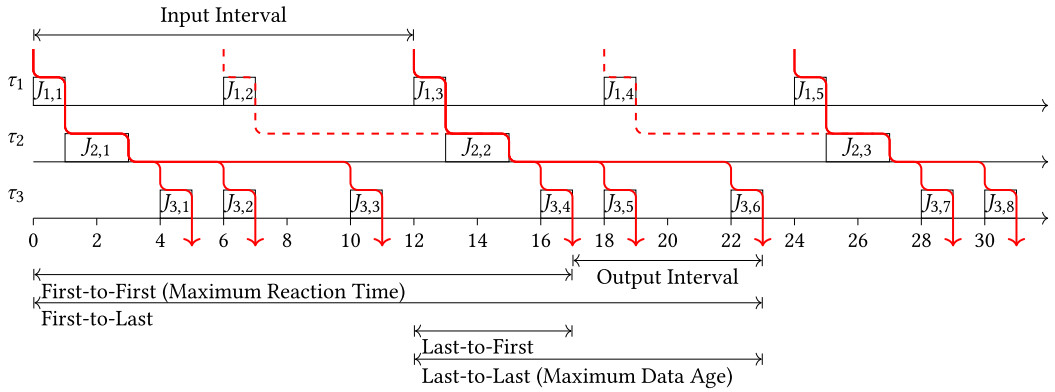

Fig. 6. End-to-end latency semantics formulated in the work of Feiertag et al. [14]. Arrows indicate data flow. Dashed lines indicate that data is overwritten before being read by the subsequent task.

Note that according to this definition, one job can be part of multiple job chains. For instance, in Figure 6, job $J_{2,3}$ is part of the job chains $(J_{1,4}, J_{2,3}, J_{3,7})$, $(J_{1,4}, J_{2,3}, J_{3,8})$, $(J_{1,5}, J_{2,3}, J_{3,7})$, and $(J_{1,5}, J_{2,3}, J_{3,8})$. The *length* $\ell(c)$ *of a job chain* $c = (J_1, J_2, \ldots, J_n)$ is the length of the time interval between the read-event of J_1 and the write-event of J_n (i.e., between the read-event of the first and the write-event of the last job in a chain):

$$\ell(J_1, J_2, \ldots, J_n) = \mathrm{we}(J_n) - \mathrm{re}(J_1). \tag{1}$$

4 End-to-End Latency

In this section, we introduce the notion of end-to-end latency, which describes the maximal time of a dataflow in a cause-effect chain $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$. Specifically, we restate different timing metrics for end-to-end latency from the literature, and introduce the definition from our previous work (cf. [19]). Analytical literature results for bounding those metrics are discussed in Section 6.

One of the first specifications of end-to-end latency was by Davare et al. [10] in 2007, where they analyzed the dataflow from a source node (left node in Figure 3) to a sink node (right node in Figure 3). Feiertag et al. [14] extended the work of Davare et al. [10] with a focus on the cases in which undersampling and oversampling occurs. They define the MRT and the MDA as the two key metrics for end-to-end analysis.

Figure 6 illustrates the end-to-end latency model from Feiertag et al. [14]. For each job J of τ_n, a *source job* of τ_1 can be determined. In other words, the job J' of τ_1 such that a data path (formally, a job chain) from J' to J exists without being overwritten. In Figure 6, the source job of $J_{3,1}$, $J_{3,2}$, and $J_{3,3}$ is $J_{1,1}$, and the source job of $J_{3,4}$, $J_{3,5}$, and $J_{3,6}$ is $J_{1,3}$. Since $J_{3,4}$, $J_{3,5}$, and $J_{3,6}$ all have the same data source, they all produce an output based on the same data. The interval that encompassed the write-events of jobs with the same data source is called the *output interval*. In Figure 6, the output interval is the interval $[17, 23]$. Feiertag et al. [14] further defined the *input interval* as the interval from the read-event of the previous data source to the read-event of the current data source. Intuitively, any input that can be captured by the system during this interval will be used by the system to create an output during the output interval. In Figure 6, the input interval for the data source $J_{1,3}$ is $[0, 12]$.

Based on these input and output intervals, Feiertag et al. [14] defined different metrics for the end-to-end latency, namely First-to-First, First-to-Last, Last-to-First, and Last-to-Last. Two key metrics are emphasized:

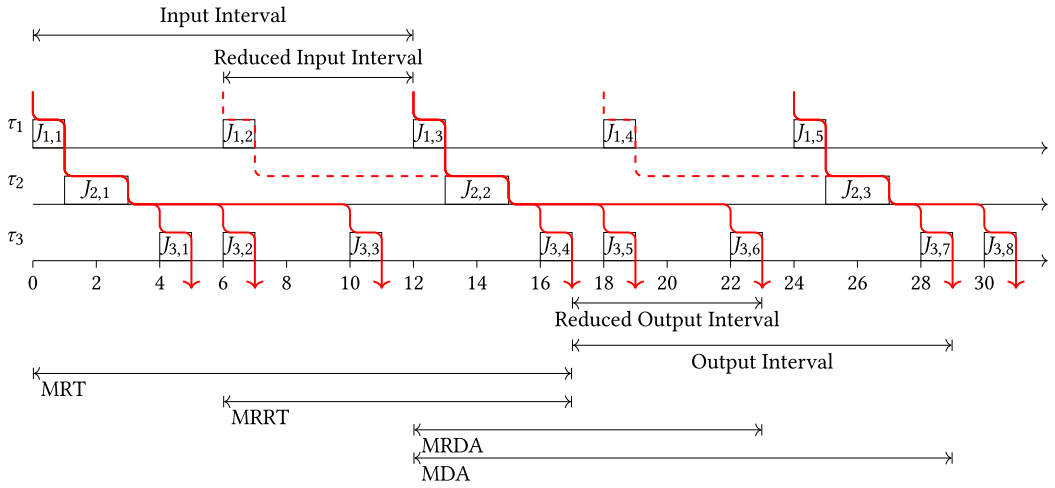

Fig. 7. Updated end-to-end latency definitions without imbalance in the input and output intervals. Again, arrows indicate dataflow, and dashed lines indicate that data is overwritten before being read by the subsequent task.

— *Maximum reaction time*: MRT is the maximum time between the beginning of the input interval and the beginning of the output interval for any data source. This is referred to as the *First-to-First* latency.
— *Maximum data age*: MDA is the maximum time between the end of the input interval and the end of the output interval for any data source. This is referred to as the *Last-to-Last* latency.

However, there is an imbalance in the definition of the input and output intervals by Feiertag et al. [14]. Specifically, the input interval is formulated as a *passive* arrival of data to the system, whereas the output interval is formulated as an *active* generation of data by the system. This imbalance is resolved by Günzel et al. [17, 18] by also considering the interval where output data that is (passively) provided by an actuator can be retrieved from the system. In other words, we extend the output interval until the write-event that originates from different input data. When applying the solution to the scenario in Figure 6, the output interval will be extended from $[17, 23]$ to $[17, 29]$.

In this tutorial, we call the output interval defined in the work from Feiertag et al. [14] the *reduced output interval* to distinguish from the proposed output interval of [17, 18]. The new definitions are depicted in Figure 7. Whereas MDA uses the extended output interval, we define the **Maximum Reduced Data Age (MRDA)** in terms of the reduced output interval. For symmetry, we also define a *reduced input interval*, which is the interval of all read-events during the input interval. In Figure 7, the reduced input interval is $[6, 12]$. Similarly, we define the **Maximum Reduced Reaction Time (MRRT)** [19]. This definition of MRT, MDA, MRRT, and MRDA removes the imbalance between the input interval and output interval, and allows the differentiation between the active and passive input and output of data. The differentiation between the MDA and MRDA, following previous work [17–19], can be summarized as follows:

— We denote the MRDA as the Last-to-Last latency from Feiertag et al. [14].
— We denote the MDA as the Last-to-Last latency with our extended output interval, covering all the time points until a write-event of a job of the last task in the cause-effect chain that originates from different input data.

Please note that whereas several literature results (e.g., [3, 4, 13]) refer to the MRDA as the MDA, we rely on the proposed more balanced definition for the tutorial.

Dürr et al. [13] suggest determining MRT and MRDA of sporadic tasks under implicit communication using *immediate forward* and *immediate backward* job chains. We reformulate their definitions in terms of more general tasks with read- and write-events, similar to prior work [19].

Definition 4.1 (Immediate forward job chain). A job chain $c = (J_1, J_2, \ldots, J_n)$ for $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$ is *immediate forward* if for all $i \in \{2, \ldots, n\}$ the job J_i is the *earliest* job of task τ_i with read-event no earlier than the write-event of J_{i-1}. In other words, J_i is the earliest job that fulfills the properties from Definition 3.1.

Definition 4.2 (Immediate backward job chain). A job chain $c = (J_1, J_2, \ldots, J_n)$ for $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$ is *immediate backward* if for all $i \in \{n-1, \ldots, 1\}$ the job J_i is the *latest* job of task τ_i with write-event no later than the read-event of J_{i+1}. In other words, J_i is the latest job that fulfills the properties from Definition 3.1.

Immediate forward job chains can be used to measure the time between the read-event in a reduced input interval and the first write-event in the corresponding output interval. For example, in Figure 7, $(J_{1,2}, J_{2,2}, J_{3,4})$ and $(J_{1,3}, J_{2,2}, J_{3,4})$ are both immediate forward job chains. Hence, comparing all immediate forward job chains yields $\mathrm{MRRT}(E) = \sup\{\ell(c) \mid c\ immediate forward job chain\}$. Similarly, immediate backward job chains are used to measure the time between the data source and a write-event in the reduced output interval. Hence, $\mathrm{MRDA}(E) = \sup\{\ell(c) \mid c\ immediate backward job chain\}$.

To cover the full time interval of the input and output interval, we extend the definition of the immediate forward and immediate backward job chains as presented in previous work [17, 18]. We refer to them as the immediate forward and immediate backward *augmented* job chains, respectively.

Definition 4.3 (Immediate forward augmented job chain). Let $z \in \mathbb{R}$ and $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$. We define the immediate forward augmented job chain at z by $\vec{ac}_z = (z, J_1, \ldots, J_{|E|}, z')$, where J_1 is the job of τ_1 with the earliest read-event $\mathrm{re}(J_1) \geq z$, the sequence $(J_1, \ldots, J_n)$ is an immediate forward job chain, and z' is at the write-event of J_n.

Definition 4.4 (Immediate backward augmented job chain). Let $z' \in \mathbb{R}$ and $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$. We define the immediate backward augmented job chain at z' by $\overleftarrow{ac}_{z'} = (z, J_1, \ldots, J_{|E|}, z')$, where J_n is the job of τ_n with the latest write-event $\mathrm{we}(J_n) \leq z'$, the sequence $(J_1, \ldots, J_n)$ is an immediate backward job chain, and z is at the read-event of J_1.

The length of an immediate forward or immediate backward augmented job chain $(z, J_1, \ldots, J_n, z')$ is defined as $\ell(z, J_1, \ldots, J_n, z') = z' - z$. Due to their construction, immediate forward augmented job chains can be used to measure the time between the beginning of the input and the beginning of the output interval—that is, the MRT. Similarly, immediate backward augmented job chains can be used to measure the time between the end of the input and the end of the output interval—that is, the MDA. Hence,

$$\mathrm{MRT}(E) = \sup_z \ell(\vec{ac}_z) \qquad \text{and} \qquad \mathrm{MDA}(E) = \sup_{z'} \ell(\overleftarrow{ac}_{z'}). \tag{2}$$

However, these definitions lack a specification of the beginning of time measurements. Specifically, if z' is chosen too small, then there is no data in the system that can be analyzed. Similarly, if z is chosen too small, the MRT approaches infinity. In fact, most applications are concerned with the system behavior only when the system is properly *warmed up*. In other words, some initial data paths that do not process any relevant data should be left out. The first time that jobs cover relevant data is at the first immediate backward job chain (that fully exists). We say that a task is warmed up as soon as the job of the first backward job chain is reached.

Definition 4.5 (Warm-up). Let $E = (\tau_1, \ldots, \tau_n)$ be a cause-effect chain. Let $c_W = (J_1^W, \ldots, J_n^W)$ be the *first* immediate backward job chain for E that exists. For all $i = 1, \ldots, n$, we say that task τ_i is *warmed up* at job J_i^W for E.

For the end-to-end latency, only job chains are considered when the system is properly warmed up. More specifically, we define the following.

Definition 4.6 (End-to-end latencies). Let $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$ be a cause-effect chain. The MRT and the MDA are defined by

$$\mathrm{MRT}(E) = \sup_{z > \mathrm{re}(J_1^W)} \ell(\vec{ac}_z) \qquad \text{and} \qquad \mathrm{MDA}(E) = \sup_{z' > \mathrm{we}(J_n^W)} \ell(\overleftarrow{ac}_{z'}). \tag{3}$$

Similarly, the MRRT and MRDA are defined by considering only the corresponding job chains without the additional z and z':

$$\mathrm{MRRT}(E) = \sup \left\{ \ell(J_1, \ldots, J_n) \,\middle|\, \vec{ac}_z = (z, J_1, \ldots, J_n, z'),\ z > \mathrm{re}\left(J_1^W\right) \right\}, \tag{4}$$

$$\mathrm{MRDA}(E) = \sup \left\{ \ell(J_1, \ldots, J_n) \,\middle|\, \overleftarrow{ac}_{z'} = (z, J_1, \ldots, J_n, z'),\ z' > \mathrm{we}\left(J_n^W\right) \right\}. \tag{5}$$

In the following, we clarify how Definition 4.6 applies to an exemplary schedule depicted in Figure 7 to determine end-to-end latencies based on job chains.

Example 4.7. We consider an example system $\mathbb{T} = \{\tau_1, \tau_2, \tau_3\}$ of three tasks with schedule depicted in Figure 7 and cause-effect chain $E = (\tau_1 \rightarrow \tau_2 \rightarrow \tau_3)$. This cause-effect chain can describe, for example, a simplified automatic brake system as depicted in Figure 1.

In Figure 7, the first immediate backward job chain is $(J_{1,1}, J_{2,1}, J_{3,1})$. Therefore, task τ_1 is warmed up at job $J_{1,1}$ for cause-effect chain E. Considering all immediate forward augmented job chains $\vec{ac}_z$ with $z > \mathrm{re}(J_{1,1})$, the largest length is achieved if z approaches $\mathrm{re}(J_{1,1})$. More specifically, for $z = 0 + \varepsilon$ with $0 < \varepsilon < 6$, we construct the immediate forward augmented job chain $\vec{ac}_z = (\varepsilon, J_{1,2}, J_{2,2}, J_{3,4}, 17)$. Hence, $\ell(\vec{ac}_z) = 17 - \varepsilon \rightarrow 17$ for $\varepsilon \rightarrow 0$. Therefore, $\mathrm{MRT}(E) = 17$. Furthermore, the longest immediate forward job chain is $(J_{1,2}, J_{2,2}, J_{3,4})$, and $\mathrm{MRRT}(E) = \ell((J_{1,2}, J_{2,2}, J_{3,4})) = \mathrm{we}(J_{3,4}) - \mathrm{re}(J_{1,2}) = 17 - 6 = 11$.

Similarly, the MDA is achieved by the immediate backward augmented job chain $\overleftarrow{ac}_{z'}$ with z' approaching 29 from below. Specifically, for $z' = 29 - \varepsilon$ with $0 < \varepsilon < 6$, we construct $\overleftarrow{ac}_{z'} = (12, J_{1,3}, J_{2,2}, J_{3,6}, 29 - \varepsilon)$. We obtain $\ell(\overleftarrow{ac}_{z'}) = 29 - \varepsilon - 12 \rightarrow 17$ for $\varepsilon \rightarrow 0$, and $\mathrm{MDA}(E) = 17$. Furthermore, the longest immediate backward job chain is $(J_{1,3}, J_{2,2}, J_{3,6})$, and $\mathrm{MRDA}(E) = \mathrm{we}(J_{3,6}) - \mathrm{re}(J_{1,3}) = 23 - 12 = 11$.

5 Fundamental Properties of End-to-End Latency

In this section, we detail two fundamental properties of the end-to-end latency. The first is a compositional property that allows to cut any cause-effect chain into smaller segments and analyze each of them separately. The second is the equivalence between the two basic metrics MRT and MDA. Although those two metrics were believed to be inherently different, it can be shown that they are actually the same. The mentioned properties are fundamental because they hold for any communication and scheduling mechanism that adheres to the two requirements stated in Section 3. Hence, it covers periodic and sporadic tasks under implicit communication and LET, and even tasks scheduled and communicating under more specialized mechanisms, like ROS2.

5.1 Compositional Property

Usually, cause-effect chains are distributed over several components that require different treatments and analyses. The analysis of such potentially long cause-effect chains can be difficult and

time consuming. To mitigate this problem, Günzel et al. [18] proved that the end-to-end latency underlies a compositional property. More specifically, any cause-effect chain can be decomposed into smaller cause-effect chains, and adding up the end-to-end latency of all components yields a safe upper bound on the end-to-end latency of the full chain. After the decomposition of the cause-effect chain, the components only have to respect the local properties that they are executed on. Hence, by a smart decomposition, the chain becomes much easier to analyze.

Theorem 5.1 (Cutting [18, Theorem 12]). *Let $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$ be any cause-effect chain, decomposed into two components $E_1 = (\tau_1 \rightarrow \cdots \rightarrow \tau_k)$ and $E_2 = (\tau_{k+1} \rightarrow \cdots \rightarrow \tau_n)$. Then the end-to-end latency of the components E_1 and E_2 yields an upper bound on the end-to-end latency of E:*

$$\mathrm{MRT}(E) \leq \mathrm{MRT}(E_1) + \mathrm{MRT}(E_2), \tag{6}$$

$$\mathrm{MDA}(E) \leq \mathrm{MDA}(E_1) + \mathrm{MDA}(E_2). \tag{7}$$

Although the theorem is only formulated for two components, applying the theorem several times allows a decomposition into more components. More specifically, if E can be decomposed into $E_1, \ldots, E_N$, then

$$\mathrm{MRT}(E) \leq \sum_{j=1}^{N} \mathrm{MRT}(E_j) \qquad \text{and} \qquad \mathrm{MDA}(E) \leq \sum_{j=1}^{N} \mathrm{MDA}(E_j). \tag{8}$$

In the work of Günzel et al. [18], the concept of *warm-up* is not established yet. Instead, they utilize so-called *valid* job chains similarly to constraint the observation window. However, using the definition of MRT and MDA with warm-up, as it is done in this tutorial, the compositional property has to be formally proven. This can be done similarly to the proof in prior work [18]. For completeness, the extended proof is provided in the appendix.

5.2 Equivalence of MRT and MDA

In the literature, MRT and MDA were analyzed independently, as they were believed to be inherently different. Starting in 2019, the analytical relation between MRT and MDA has been further explored. More specifically, Dürr et al. [13] showed that an analytical bound for MRT in sporadic systems also holds for the MRDA. Furthermore, Günzel et al. [17, 18] showed that the exact MRT is an upper bound for the MDA under a general definition.

Empirical observations suggest an even stronger relation. More specifically, the AUTOSAR Timing Extensions [2] provide an important observation about the relation between MRT and MDA, namely that "without over- and undersampling, age and reaction are the same" [2, Section 3.6.2, p. 114]. While this observation seems to imply that MRT and MDA can differ for systems with over- or undersampling, recent measurements in ROS2 show that the observed MRT and MDA always coincide [41]. Both observations suggest a strong relation between MRT and MDA, but no proof for such a relation has been provided. Recently, Günzel et al. [19] showed that MRT and MDA are actually just two descriptions with different perspective of the same end-to-end latency.

Theorem 5.2 (Equivalence [19, Theorem 14]). *The MRT and the MDA are equivalent. More specifically,*

$$\mathrm{MRT}(E) = \mathrm{MDA}(E) \tag{9}$$

for any cause-effect chain E.

Due to the equivalence, no distinction between MRT and MDA is necessary. Hence, in the following, we use the term

$$\mathrm{Lat}(E) := \mathrm{MRT}(E) = \mathrm{MDA}(E) \tag{10}$$

for the general end-to-end latency. The authors generalize the description of the end-to-end latency even further by introducing p-partitioned job chains. While for immediate forward augmented job chains $\vec{ac}_z$ the chain is generated from the beginning, and for immediate backward augmented job chains $\overleftarrow{ac}_z$ the chain is generated from the end, p-partitioned job chains pc_m^p are generated from the p-th task. The authors show that

$$\mathrm{Lat}(E) = \sup_m \ell\left(pc_m^p\right) \tag{11}$$

for any $p \in \{1, \ldots, n\}$. Further details on the construction and definition of p-partitioned job chains can be found in the work by Günzel et al. [19].

The equivalence result can be transferred to the MRRT and the MRDA as well.[1]

Theorem 5.3 (Relation with MRRT and MRDA [19, Theorem 18]). *Let $\rho^-(\tau)$ and $\rho^+(\tau)$ be the minimal and maximal time between two subsequent read-events of a task τ, respectively. Furthermore, let $\psi^-(\tau)$ and $\psi^+(\tau)$ be the minimal and maximal time between two subsequent write-events of a task τ, respectively. Then*

$$\mathrm{Lat}(E) - \mathrm{MRRT}(E) \in [\rho^-(\tau_1), \rho^+(\tau_1)] \tag{12}$$

$$\mathrm{Lat}(E) - \mathrm{MRDA}(E) \in [\psi^-(\tau_n), \psi^+(\tau_n)] \tag{13}$$

for any cause-effect chain $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$.

This relation is particularly interesting for tasks underlying the LET communication mechanism. More specifically, it shows that $\mathrm{MRRT}(E) + T_{\tau_1} = \mathrm{Lat}(E) = \mathrm{MRDA}(E) + T_{\tau_n}$ if all tasks in E adhere to the LET communication mechanism.

6 Analytical Results

Starting from 2007, and more frequently since 2016, several analytical bounds for the end-to-end latency have been provided in the literature. Table 1 provides an overview over these analytical results. We note that the analyses may have different assumptions for the system design that have to be carefully checked when applying an analysis from the table. The analytical approaches for periodic and sporadic systems are usually different. Periodic approaches mostly traverse a safe analysis window (commonly one or two hyperperiods) and quantify the length of job chains starting in that analysis window. Prime examples are the analytical results from Becker et al. [3, 4] for MRDA and Kloda et al. [29] for MRT. Recent approaches focus on more dedicated solutions [16, 18, 18, 30, 37] and on speeding up the computation process [5, 19]. Due to their complexity, we refer to the literature for further information on bounds for periodic tasks.

However, sporadic schedules are not repetitive. Therefore, it is impossible to define a safe analysis window. Approaches for sporadic systems rather pursue closed-form solutions with analytical overapproximation. To that end, the first analytical bound on the end-to-end latency was provided by Davare et al. [10] under implicit communication. Although the bound was originally formulated for periodic tasks, the analysis can be applied to sporadic tasks with maximum inter-arrival time T_τ^{max} as well.

Theorem 6.1 (Davare et al. [10, Section 2.1]). *If tasks of the cause-effect chain $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$ communicate via implicit communication, then the end-to-end latency* $\mathrm{Lat}(E)$ *is upper*

[1] Please note that the formulation in the work of Günzel et al. [19, Theorem 18] has a typo, using distance between read-events instead of distance between write-events for the MRDA, which has no impact on the remaining results of Günzel et al. [19]. Our formulation in Theorem 5.3 shows the correct formulation.

Table 1. Analytical Results for End-to-End Latency in the Literature

Year	Paper	E2E Latency	Communication Policy	Release Policy
2007	Davare et al. [10]	MRT	implicit	sporadic*
2007	Natale et al. [36]	MRT	implicit	other
2012	Zhu et al. [44]	MRRT	implicit	periodic
2016	Becker et al. [3]	MRDA	implicit	periodic
2017	Becker et al. [4]	MRDA	explicit, implicit, LET	periodic
2017	Hamann et al. [22]	MRT	implicit, LET	periodic, sporadic*
2018	Kloda et al. [29]	MRT	implicit	periodic
2019	Dürr et al. [13]	MRDA, MRT	implicit	sporadic
2020	Kordon and Tang [32]	MRDA	LET	periodic
2020	Martinez et al. [35]	MRDA, MRT	explicit, implicit, LET	periodic
2021	Günzel et al. [17]	MRDA, MDA, MRT	implicit	periodic
2022	Teper et al. [41]	MDA, MRT	implicit	other
2022	Gohari et al. [16]	MRDA	implicit	periodic
2022	Kloda et al. [30]	MRT	implicit	periodic
2022	Pazzaglia and Maggio [37]	MRDA	LET	periodic
2022	Bi et al. [5]	MRDA	implicit	periodic
2023	Günzel et al. [20]	MRT	implicit, LET	periodic, sporadic
2023	Tang et al. [39]	MRT	implicit	other
2023	Günzel et al. [19]	MRDA, MDA, MRRT, MRT	LET	periodic
2023	Günzel et al. [18][2]	MRDA, MDA, MRT	implicit	periodic

*Although formulated for periodic tasks, their analysis is applicable to sporadic tasks as well.
Please note that although MRT and MDA are shown to be equivalent [19], they are distinguished here to clarify the intentions of the authors.

bounded by

$$\mathrm{Lat}(E) \leq \sum_{i=1}^{n} T_{\tau_i}^{max} + R_{\tau_i} \tag{14}$$

where R_i is the worst-case response time of task τ_i.

The bound was further tightened and extended to the MRDA by Dürr et al. [13] in 2019. Their bound on the end-to-end latency can be summarized as follows.

Theorem 6.2 (Dürr et al. [13, Theorem 5.4]). *If tasks of the cause-effect chain $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$ communicate via implicit communication, then the end-to-end latency* $\mathrm{Lat}(E)$ *is upper bounded by*

$$\mathrm{Lat}(E) \leq \sum_{i=1}^{n} T_{\tau_i}^{max} + R_{\tau_i} - \sum_{i=1}^{n-1} [P_i] \cdot \min\left(R_{\tau_i}, T_{\tau_{i+1}}^{max}\right) \tag{15}$$

where $[P_i] = 1$ if τ_i and τ_{i+1} run on the same processor and τ_{i+1} has lower priority than τ_i, and $[P_i] = 0$ otherwise.

For tasks under LET communication, Hamann et al. [22] provided an upper bound for the end-to-end latency. Although their paper is formulated for periodic tasks, this particular bound is valid for sporadic tasks as well.

Theorem 6.3 (Hamann et al. [22, Section 4.1.3]). *If tasks of the cause-effect chain $E = (\tau_1 \rightarrow \cdots \rightarrow \tau_n)$ communicate under LET, then the end-to-end latency* $\mathrm{Lat}(E)$ *is upper bounded by*

[2]This work is a journal extension of a conference paper [17].

$$\mathrm{Lat}(E) \leq \sum_{i=1}^{n} T_{\tau_i}^{max} + D_{\tau_i}. \tag{16}$$

The work by Günzel et al. [20] studies heterogeneous systems where periodic and sporadic tasks as well as implicit communication and LET communication are present in tasks of the same cause-effect chain. To that end, they uncover the principles that most state-of-the-art analytical results are built from. Their analytical framework can be utilized to prove most results from the literature in a unified manner.

Analysis with Alternative System Models. While the end-to-end behavior of systems with scheduling mechanisms apart from typical time-triggered fixed-priority scheduling has been explored already in 2007 (e.g., by Natale et al. [36]), recently this research direction has gained special interest. Specifically, Teper et al. [41] provide an end-to-end analysis for systems using ROS2, which allows both event-triggered and time-triggered release mechanisms. Dasari et al. [9] formally describe the real-time behavior of the Adaptive Partitioning Scheduler and develop an end-to-end latency bound for event chains under the Adaptive Partitioning Scheduler. Tang et al. [39] propose a third option to trigger the processing tasks in a chain, namely event-triggered with data refreshing, and provide an end-to-end analysis for that case. Moreover, Tang et al. [40] compare different communication paradigms regarding the end-to-end latency of cause-effect chains (namely, implicit communication, Dynamic Buffering Protocol, and LET), and discuss the impact of priority assignment on end-to-end latency.

System Design. The key problem from the design perspective (i.e., the system configuration to meet end-to-end timing requirements) has already been studied in 2007 by Davare et al. [10]. In their work, and in related work by Natale et al. [36], Zhu et al. [44], Schlatow et al. [38], and Wang et al. [43], the task configuration is optimized to meet end-to-end requirements. Further mechanisms that shape the system for advantageous behavior of end-to-end latencies are presented by Hong et al. [24], Klaus et al. [28], and Bini et al. [6]. Klaus et al. [28] combine the potential of job-level optimization with the determinism and low overheads of static, task-level approaches. To that end, data-age constrained task sets with job-level dependencies are mapped on event-triggered systems. Bini et al. [6] introduce a model to capture the behavior of a producer-consumer pair of tasks. Using ring algebra, the combined behavior of the pair can be modeled as a single periodic task. Their work further presents a lightweight mechanism to eliminate jitter in a chain of any size, resulting in a single periodic LET task with zero jitter.

Related End-to-End Metrics. While end-to-end latency of cause-effect chains usually focuses on individual paths of data through the system, dataflow models provide a more holistic view on the propagation of data packets [1]. An important metric for such dataflow models is the flow completion time [12, 24]. Other metrics related to the end-to-end latency have been presented and discussed in 2023. Köhler et al. [31] define *robustness margins* which guarantee that if software extensions stay within certain bounds, then the end-to-end deadline of a cause-effect chain can still be satisfied. Günzel et al. [21] define and analyze *probabilistic reaction time* considering two types of randomness: response time randomness and failure probabilities. Jiang et al. [25] analyze and optimize of the *worst-case time disparity* (i.e., the maximum difference among the timestamps of all raw data produced by sensors that an output originates from) in cause-effect chains.

7 Summary and Open Problems

This tutorial provides a systematic picture of end-to-end analysis for cause-effect chains. It is our hope that it serves as a foundation for understanding this evolving area and can be utilized as lecture notes for students and lecturers. As shortly summarized at the end of Section 6, there have

been many exciting results reported in the recent years. Furthermore, we identify the following research directions that may be interesting to explore:

— *Configuration of system parameters to meet requirements*: This tutorial provides fundamental end-to-end timing analyses, assuming that the periods (or inter-arrival times) of recurrent tasks are specified and the priority assignments are given. Essentially, such parameters have to be decided by the system designer, who may not be aware of the best configurations. There have been some results to find such optimized configurations in the literature (e.g., [10, 38, 43]), but the fundamental properties for such optimization have not yet been fully revealed. It is highly demanded to explore such fundamental properties.

— *Probabilistic end-to-end latency*: The observed end-to-end latency may be much lower than the worst case, which leads to the motivation of probabilistic end-to-end latency—that is, probabilistic reaction time and data age. This research direction is highly motivated by the recent development of probabilistic reasoning on the response times of real-time systems (e.g., [7, 8, 11, 34, 42, 42]). Although this topic is highly promising, it has been explored only to a limited extent so far. Specifically, to date there is only one result for probabilistic reaction time [21].

— *Interplay and merging of multiple chains*: While the typical definition of end-to-end latency only considers a single cause-effect chain, the data dependencies present in real applications are much more complex. Specifically, typical systems comprise multiple chains that merge or have interplay with each other. While evolving metrics like the time disparity [25] define specific problems that arise from the use of multiple chains, we believe that further studies are needed to determine the fundamental timing properties of systems with multiple interacting cause-effect chains.

Appendix

A Proof of Theorem 5.1

We start by proving Equation (6). Consider an immediate forward augmented job chain $\vec{ac}_z = (z, J_1, \ldots, J_n, \mathrm{we}(J_n))$ for E. We define the immediate forward augmented job chains $\vec{ac}^1_z = (z, J_1, \ldots, J_k, \mathrm{we}(J_k))$ for E_1 and $\vec{ac}^2_{\mathrm{we}(J_k)} = (\mathrm{we}(J_k), J_{k+1}, \ldots, J_n, \mathrm{we}(J_n))$ for E_2. Then $\ell(\vec{ac}_z) \leq \ell(\vec{ac}^1_z) + \ell(\vec{ac}^2_{\mathrm{we}(J_k)})$. We must check that if τ_1 is warmed up at time z for E, then (1) τ_1 is warmed up at time z for E_1, and (2) τ_{k+1} is warmed up at time $\mathrm{we}(J_k)$ for E_2.

If τ_1 is warmed up at time z for E, then there exists an immediate backward job chain $\overleftarrow{c} = (\tilde{J}_1, \ldots, \tilde{J}_n)$ for E such that $\mathrm{re}(\tilde{J}_1) < z$. In that case $(\tilde{J}_1, \ldots, \tilde{J}_k)$ is an immediate backward job chain for E_1 with $\mathrm{re}(\tilde{J}_1) < z$. Hence, (1) holds.

We define the immediate backward job chain $(\tilde{J}_{k+1}, \ldots, \tilde{J}_n)$ for E_2 and show that $\mathrm{re}(\tilde{J}_{k+1}) < \mathrm{we}(J_k)$ by contradiction. To that end, assume that $\mathrm{re}(\tilde{J}_{k+1}) \geq \mathrm{we}(J_k)$. Since $\tilde{J}_k$ is the job with the maximal write-event such that $\mathrm{we}(\tilde{J}_k) \leq \mathrm{re}(\tilde{J}_{k+1})$, we know that $\mathrm{we}(J_k) \leq \mathrm{we}(\tilde{J}_k)$. In that case, $\mathrm{re}(J_k) \leq \mathrm{re}(\tilde{J}_k)$ and also $\mathrm{re}(J_1) \leq \mathrm{re}(\tilde{J}_1)$ since $\overleftarrow{c}$ is immediate backward. This contradicts $\mathrm{re}(J_1) > \mathrm{re}(\tilde{J}_1)$. Hence, (2) holds as well.

We conclude $\ell(\vec{ac}_z) \leq \ell(\vec{ac}^1_z) + \ell(\vec{ac}^2_{\mathrm{we}(J_k)}) \leq \mathrm{MRT}(E_1) + \mathrm{MRT}(E_2)$. Since this holds for all z such that τ_1 is warmed up for E at z, we obtain $\mathrm{MRT}(E) \leq \mathrm{MRT}(E_1) + \mathrm{MRT}(E_2)$. □ Equation (6)

Equation (7) is proven analogously. Here, the length of any immediate backward augmented job chain $\overleftarrow{ac}_z = (\mathrm{re}(J_1), J_1, \ldots, J_n, z)$ for the cause-effect chain E is upper bounded by $\ell(\overleftarrow{ac}_z) \leq \ell(\overleftarrow{ac}^1_{\mathrm{re}(J_{k+1})+\varepsilon}) + \ell(\overleftarrow{ac}^2_z)$, where $\overleftarrow{ac}^1_{\mathrm{re}(J_{k+1})+\varepsilon} = (\mathrm{re}(J_1), J_1, \ldots, J_k, \mathrm{re}(J_{k+1}) + \varepsilon)$ with $0 < \varepsilon$ sufficiently small is an immediate backward augmented job chain for E_1 and $\overleftarrow{ac}^2_z = (\mathrm{re}(J_{k+1}), J_{k+1}, \ldots, J_n, z)$ is an immediate backward augmented job chain for E_2.

If τ_n is warmed up at z for E, then there exists an immediate backward job chain $\overleftarrow{c} = (\tilde{J}_1, \ldots, \tilde{J}_n)$ for E such that $z > \mathrm{re}(\tilde{J}_n)$. In that case, $\mathrm{re}(J_n) \geq \mathrm{re}(\tilde{J}_n)$. Therefore, $\mathrm{re}(J_i) \geq \mathrm{re}(\tilde{J}_i)$ for all $i = 1, \ldots, n$ because $\overleftarrow{c}$ is immediate backward. The exists the immediate backward job chain $(\tilde{J}_1, \ldots, \tilde{J}_k)$ for E_1 satisfies $\mathrm{re}(\tilde{J}_k) \leq \mathrm{we}(\tilde{J}_k) \leq \mathrm{re}(\tilde{J}_{k+1}) < \mathrm{re}(J_{k+1}) + \varepsilon$. Moreover, the immediate backward job chain $(\tilde{J}_{k+1}, \ldots, \tilde{J}_n)$ for E_2 satisfies $\mathrm{re}(\tilde{J}_n) < z$. Hence, τ_k is warmed up at $\mathrm{re}(J_{k+1}) + \varepsilon$ for E_1 and τ_n is warmed up at z for E_2. Therefore, we obtain $\ell(\overleftarrow{ac}_z) \leq \ell(\overleftarrow{ac}^1_{\mathrm{re}(J_{k+1})}) + \ell(\overleftarrow{ac}^2_z) \leq \mathrm{MDA}(E_1) + \mathrm{MDA}(E_2)$.Since this holds for all z such that τ_n is warmed up for E at z, we obtain $\mathrm{MDA}(E) \leq \mathrm{MDA}(E_1) + \mathrm{MDA}(E_2)$. □ Equation (7)

References

[1] Tyler Akidau, Robert Bradshaw, Craig Chambers, Slava Chernyak, Rafael Fernández-Moctezuma, Reuven Lax, Sam McVeety, Daniel Mills, Frances Perry, Eric Schmidt, et al. 2015. The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proc. VLDB Endow.* 8, 12 (2015), 1792–1803.

[2] AUTOSAR. 2022. Specification of Timing Extensions (AUTOSAR CP R22-11). Retrieved January 18, 2024 from https://www.autosar.org/fileadmin/standards/R22-11/CP/AUTOSAR_TPS_TimingExtensions.pdf

[3] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. 2016. Synthesizing job-level dependencies for automotive multi-rate effect chains. In *Proceedings of the International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'16)*. 159–169. https://doi.org/10.1109/RTCSA.2016.41

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

[5] Ran Bi, Xinbin Liu, Jiankang Ren, Pengfei Wang, Huawei Lv, and Guozhen Tan. 2022. Efficient maximum data age analysis for cause-effect chains in automotive systems. In *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC'22)*. ACM, 1243–1248.

[6] Enrico Bini, Paolo Pazzaglia, and Martina Maggio. 2023. Zero-jitter chains of periodic LET tasks via algebraic rings. *IEEE Trans. Comput.* 72, 11 (2023), 3057–3071.

[7] Sergey Bozhko, Georg von der Brüggen, and Björn B. Brandenburg. 2021. Monte Carlo response-time analysis. In *Proceedings of the 42nd IEEE Real-Time Systems Symposium (RTSS'21)*. IEEE, 342–355. https://doi.org/10.1109/RTSS52674.2021.00039

[8] Kuan-Hsun Chen, Mario Günzel, Georg von der Brüggen, and Jian-Jia Chen. 2022. Critical instant for probabilistic timing guarantees: Refuted and revisited. In *Proceedings of the 43rd IEEE Real-Time Systems Symposium (RTSS'22)*. 145–157. https://doi.org/10.1109/RTSS55097.2022.00022

[9] Dakshina Dasari, Matthias Becker, Daniel Casini, and Tobias Blaß. 2022. End-to-end analysis of event chains under the QNX adaptive partitioning scheduler. In *Proceedings of the 28th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'22)*. IEEE, 214–227.

[10] Abhijit Davare, Qi Zhu, Marco Di Natale, Claudio Pinello, Sri Kanajan, and Alberto L. Sangiovanni-Vincentelli. 2007. Period optimization for hard real-time distributed automotive systems. In *Proceedings of the Design Automation Conference (DAC'07)*. 278–283. https://doi.org/10.1145/1278480.1278553

[11] Robert I. Davis and Liliana Cucu-Grosjean. 2019. A survey of probabilistic schedulability analysis techniques for real-time systems. *Leibniz Trans. Embed. Syst.* 6, 1 (2019), Article 4, 53 pages. https://doi.org/10.4230/LITES-v006-i001-a004

[12] Nandita Dukkipati and Nick McKeown. 2006. Why flow-completion time is the right metric for congestion control. *Comput. Commun. Rev.* 36, 1 (2006), 59–62.

[13] Marco Dürr, Georg von der Brüggen, Kuan-Hsun Chen, and Jian-Jia Chen. 2019. End-to-end timing analysis of sporadic cause-effect chains in distributed systems. *ACM Trans. Embed. Comput. Syst.* 18, 5s (2019), Article 58, 24 pages. https://doi.org/10.1145/3358181

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

[15] David Gay, Philip Alexander Levis, J. Robert von Behren, Matt Welsh, Eric A. Brewer, and David E. Culler. 2003. The *nesC* language: A holistic approach to networked embedded systems. *ACM SIGPLAN Notices* 38, 5 (2003), 1–11.

[16] Pourya Gohari, Mitra Nasri, and Jeroen Voeten. 2022. Data-age analysis for multi-rate task chains under timing uncertainty. In *Proceedings of the 30th International Conference on Real-Time Networks and Systems (RTNS'22)*. ACM, 24–35.

[17] Mario Günzel, Kuan-Hsun Chen, Niklas Ueter, Georg von der Brüggen, Marco Dürr, and Jian-Jia Chen. 2021. Timing analysis of asynchronized distributed cause-effect chains. In *Proceedings of the 27th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'21)*. IEEE, 40–52.

[18] Mario Günzel, Kuan-Hsun Chen, Niklas Ueter, Georg von der Brüggen, Marco Dürr, and Jian-Jia Chen. 2023. Compositional timing analysis of asynchronized distributed cause-effect chains. *ACM Trans. Embed. Comput. Syst.* 22, 4 (2023), Article 63, 34 pages.

[19] Mario Günzel, Harun Teper, Kuan-Hsun Chen, Georg von der Brüggen, and Jian-Jia Chen. 2023. On the equivalence of maximum reaction time and maximum data age for cause-effect chains. In *35th Euromicro Conference on Real-Time Systems (ECRTS 2023)*. Leibniz International Proceedings in Informatics, Vol. 262. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, Article 10, 22 pages.

[20] Mario Günzel, Niklas Ueter, Kuan-Hsun Chen, and Jian-Jia Chen. 2023. Timing analysis of cause-effect chains with heterogeneous communication mechanisms. In *Proceedings of the 31st International Conference on Real-Time Networks and Systems (RTNS'23)*. ACM, 224–234.

[21] Mario Günzel, Niklas Ueter, Kuan-Hsun Chen, Georg von der Brüggen, and Jian-Jia Chen. 2023. Probabilistic reaction time analysis. *ACM Trans. Embed. Comput. Syst.* 22, 5s (2023), Article 143, 22 pages.

[22] Arne Hamann, Dakshina Dasari, Simon Kramer, Michael Pressler, and Falk Wurst. 2017. Communication centric design in complex automotive embedded systems. In *Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS'17)*. Article 10, 20 pages.

[23] Thomas A. Henzinger, Benjamin Horowitz, and Christoph M. Kirsch. 2003. Giotto: A time-triggered language for embedded programming. *Proc. IEEE* 91, 1 (2003), 84–99.

[24] Chi-Yao Hong, Matthew Caesar, and Brighten Godfrey. 2012. Finishing flows quickly with preemptive scheduling. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'12)*. ACM, 127–138.

[25] Xu Jiang, Xiantong Luo, Nan Guan, Zheng Dong, Shaoshan Liu, and Wang Yi. 2023. Analysis and optimization of worst-case time disparity in cause-effect chains. In *Proceedings of the 2023 Design, Automation, and Test in Europe Conference and Exhibition (DATE'23)*. IEEE, 1–6.

[26] Mathai Joseph and Paritosh K. Pandya. 1986. Finding response times in a real-time system. *Comput. J.* 29, 5 (1986), 390–395.

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

[28] Tobias Klaus, Matthias Becker, Wolfgang Schröder-Preikschat, and Peter Ulbrich. 2021. Constrained data-age with job-level dependencies: How to reconcile tight bounds and overheads. In *Proceedings of the 27th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'21)*. IEEE, 66–79.

[29] Tomasz Kloda, Antoine Bertout, and Yves Sorel. 2018. Latency analysis for data chains of real-time periodic tasks. In *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'18)*. 360–367. https://doi.org/10.1109/ETFA.2018.8502498

[30] Tomasz Kloda, Jiyang Chen, Antoine Bertout, Lui Sha, and Marco Caccamo. 2022. Latency analysis of self-suspending task chains. In *Proceedings of the 2022 Design, Automation, and Test in Europe Conference and Exhibition (DATE'22)*. IEEE, 1299–1304.

[31] Leonie Köhler, Phil Hertha, Matthias Beckert, Alex Bendrick, and Rolf Ernst. 2023. Robust cause-effect chains with bounded execution time and system-level logical execution time. *ACM Trans. Embed. Comput. Syst.* 22, 3 (2023), Article 50, 28 pages.

[32] Alix Munier Kordon and Ning Tang. 2020. Evaluation of the age latency of a real-time communicating system using the LET paradigm. In *32nd Euromicro Conference on Real-Time Systems (ECRTS 2020)*. Leibniz International Proceedings on Informatics, Vol. 165. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, Article 20, 20 pages.

[33] John P. Lehoczky, Lui Sha, and Ye Ding. 1989. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the Real-Time Systems Symposium (RTSS'89)*. 166–171.

[34] Filip Markovic, Pierre Roux, Sergey Bozhko, Alessandro V. Papadopoulos, and Björn B. Brandenburg. 2021. CTA: A correlation-tolerant analysis of the deadline-failure probability of dependent tasks. In *Proceedings of the 42nd IEEE Real-Time Systems Symposium (RTSS'21)*.

[35] Jorge Martinez, Ignacio Sañudo, and Marko Bertogna. 2020. End-to-end latency characterization of task communication models for automotive systems. *Real Time Syst.* 56, 3 (2020), 315–347.

[36] Marco Di Natale, Wei Zheng, Claudio Pinello, Paolo Giusto, and Alberto L. Sangiovanni-Vincentelli. 2007. Optimizing end-to-end latencies by adaptation of the activation events in distributed automotive systems. In *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'07)*. IEEE, 293–302.

[37] Paolo Pazzaglia and Martina Maggio. 2022. Characterizing the effect of deadline misses on time-triggered task chains. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 41, 11 (2022), 3957–3968.

[38] Johannes Schlatow, Mischa Möstl, Sebastian Tobuschat, Tasuku Ishigooka, and Rolf Ernst. 2018. Data-age analysis and optimisation for cause-effect chains in automotive control systems. In *Proceedings of the IEEE International Symposium on Industrial Embedded Systems (SIES'18)*. 1–9.

[39] Yue Tang, Nan Guan, Xu Jiang, Zheng Dong, and Wang Yi. 2023. Reaction time analysis of event-triggered processing chains with data refreshing. In *Proceedings of the 60th ACM/IEEE Design Automation Conference (DAC'23)*. IEEE, 1–6.

[40] Yue Tang, Xu Jiang, Nan Guan, Dong Ji, Xiantong Luo, and Wang Yi. 2023. Comparing communication paradigms in cause-effect chains. *IEEE Trans. Comput.* 72, 1 (2023), 82–96.

[41] Harun Teper, Mario Günzel, Niklas Ueter, Georg von der Brüggen, and Jian-Jia Chen. 2022. End-to-end timing analysis in ROS2. In *Proceedings of the 43rd IEEE Real-Time Systems Symposium (RTSS'22)*. IEEE, 53–65.

[42] Georg von der Brüggen, Nico Piatkowski, Kuan-Hsun Chen, Jian-Jia Chen, Katharina Morik, and Björn B. Brandenburg. 2021. Efficiently approximating the worst-case deadline failure probability under EDF. In *Proceedings of the 42nd IEEE Real-Time Systems Symposium (RTSS'21)*. 214–226. https://doi.org/10.1109/RTSS52674.2021.00029

[43] Sen Wang, Dong Li, Ashrarul H. Sifat, Shaoyu Huang, Xuanliang Deng, Changhee Jung, Ryan K. Williams, and Haibo Zeng. 2023. Optimizing logical execution time model for both determinism and low latency. *CoRR abs/2310.19699* (2023).

[44] Qi Zhu, Haibo Zeng, Wei Zheng, Marco Di Natale, and Alberto L. Sangiovanni-Vincentelli. 2012. Optimization of task allocation and priority assignment in hard real-time distributed systems. *ACM Trans. Embed. Comput. Syst.* 11, 4 (2012), Article 85, 30 pages.

Received 19 January 2024; revised 9 September 2024; accepted 22 September 2024