Latency upper bound for data chains of real-time periodic tasks

Tomasz Kloda[1], Antoine Bertout[2], and Yves Sorel[3]

[1]Technische Universität München, Garching bei München, Germany, tomasz.kloda@tum.de
[2]LIAS, Université de Poitiers, ISAE-ENSMA, Poitiers, France, antoine.bertout@univ-poitiers.fr
[3]Inria de Paris, Paris 75012, France, yves.sorel@inria.fr

Abstract

The inter-task communication in embedded real-time systems can be achieved using various patterns and be subject to different timing constraints. One of the most basic communication patterns encountered in today's automotive and aerospace software is the data chain. Each task of the chain reads data from the previous task and delivers the results of its computation to the next task. The data passing does not affect the execution of the tasks that are activated periodically at their own rates. As there is no task synchronization, a task does not wait for its predecessor data; it may execute with old data and get new data at its later release. From the design stage of embedded real-time systems, evaluating if data chains meet their timing requirements, such as the latency constraint, is of the highest importance. The trade-off between accuracy and complexity of the timing analysis is a critical element in the optimization process. In this paper, we consider data chains of real-time periodic tasks executed by a fixed-priority preemptive scheduler upon a single processor. We present a method for the worst-case latency calculation of periodic tasks' data chains. As the method has an exponential time complexity, we derive a polynomial-time upper bound. Evaluations carried out on an automotive benchmark demonstrate that the average bound overestimation is less than 10 percent of the actual value.

Keywords— Real-Time Systems, End-to-End Latency, Periodic Tasks, Read-Execute-Write.

1 Introduction

The cyber-physical systems are engineered solutions that bridge together the cyber and physical world. At the boundaries of these two worlds, a connection is established through two peripheral subsystems. A sensory subsystem detects the events appearing in the environment and an actuation subsystem delivers the appropriate actions to control the physical process. Low-latency input-output processing is a desirable property for enabling time-sensitive control. The software architecture for such systems usually consists of multiple cooperating and communicating tasks that process the data from sensors to actuators. The resulting task chain end-to-end latency necessarily depends on the execution requirements of individual tasks and the inter-task communication delays.

The compound architectures of the real-time embedded systems in which the flow of data requires several processing stages before the actuation signals are produced can be encountered in many mission- and safety-critical applications. Such applications can be found in fields like aeronautics, healthcare, or automotive.

A *Flight Management System* (*FMS*) can be viewed as an example of the cyber-physical system with multiple-stage data processing where the output of one task or task group is the input of the next one. The *FMS* is an on-board avionics computing system used in new-generation aircraft that helps a pilot in operating a wide range of in-flight tasks [62]. It keeps track of the current aircraft position, assists the crew in flight planning and trajectory prediction, monitors and helps to optimize the flight parameters and performances (e.g., fuel consumption, flight time). Figure 1 gives an overview of *FMS* software architecture [1] (sensors as triangles, actuators as ellipses, and functions as boxes). As can be seen, many *FMS* functions are divided into subsequent groups that process the data from the other avionics systems and interfaces. Consider, for instance, the chain marked with the red dotted line in Figure 1. First, the readings from the autonomous sensors, like accelerometers and gyroscopes, together with the information coming from the navigation receivers, like the Distance Measuring Equipment, the VHF Omni Directional Radio Range or the Global Positioning System, are fused in by the tasks of the Sensor Group. Then, the tasks of the Localization Group, based on the information from the navigation database and the fused sensor readings, compute an accurate estimate of the current aircraft position. Afterward,

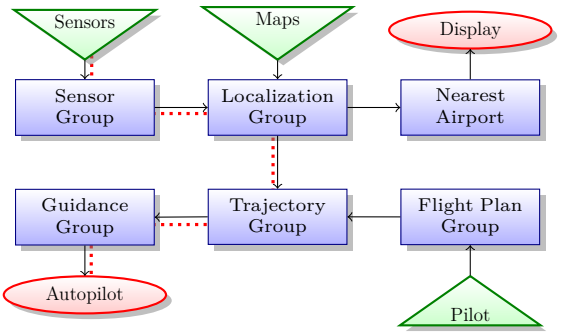


Figure 1: Flight Management System software architecture.

[1]See [18, 49] for a more detailed description. The interesting examples of the automotive systems with similar data processing structure can be found in Chapter 4 of [48] and in [8].

the current position is processed by a chain of tasks from the Trajectory Group and the Guidance Group to deliver the steering signals to the actuators.

End-to-end latency constraints. Due to the mission-critical nature of many operations mentioned in the previous *FMS* example, their results must be produced within a precise time. Wyss et al. [63] summarize different timing requirements as follows. *Reactivity* (or data reaction [45]) is a time interval during which data must be present at the sensor input to be detected. *Latency* is a time interval between the arrival of data at the sensor input and actuator's update with a value corresponding to that particular data (i.e., the time needed for data to be propagated from sensor to actuator). *Freshness* (or data age [21]) is a time interval during which the actuator's value corresponding to the same sensor reading is used.

Task chains. From sensor readings to actuation signals, the *FMS* tasks process the data through several stages. A *task chain* is a sequence of communicating tasks in which every task receives the data from its predecessors [45]. There are two models of task chains: trigger chains and data chains. They differ in task activation rules. In *trigger* chains, the tasks are released by the events issued from the preceding tasks. In *data* chains, the tasks are independent, and every task is activated periodically at its individual rate.

Trigger chains are characterized by a more flexible timing behavior. Their tasks execute only if necessary: a task is released only if it receives the request from its predecessor. The processor resources are used efficiently, and the inter-task data passing is immediate. Conversely, tasks of the data chains are released at every period, even if there is no new data to process. Processor resources can be wasted in the case of redundant task execution. Moreover, certain delay of time can elapse between the instant when the results are produced by the current task and the instant when the next task is released and can retrieve the results.

However, especially in an industrial context, data chains may outweigh the above-mentioned advantages of trigger chains. In fact, imposing the precedence relation between the subsequent tasks of a trigger chain requires synchronization mechanisms (e.g., semaphores). The use of the synchronization primitives can lead to priority-inversion problems and deadlock formations [57, 58]. An additional effort is therefore needed for the integration of these primitives with adequate control structures. As safety-critical software is subject to stringent standards, adding these features to the system makes the certification process ultimately more complicated [22]. For those reasons, the unrestrained execution model of data chains can permit to reduce cost and time overheads related to the correct implementation and validation of synchronization mechanisms required by trigger chains.

Motivation. The respect of timing constraints is crucial for critical real-time embedded systems and must be taken into account at the earliest stages of the design process. End-to-end timing constraints are introduced in many automotive software architectures and modeling languages [46] (e.g., timing model of the AUTOSAR standard [3], timing extensions to EAST-ADL [19], TADL2 language [59]). Computing a single value of data chain latency is not computationally challenging. It is especially true for the short chains and the chains with equal periods. However, already at the design level, the computational complexity becomes a serious limitation when multiple possible design choices must be evaluated. With the rapid increase of software size and complexity in today's embedded systems [10, 20], the problem can only be exacerbated. Lowering the computational complexity of the latency analysis can help in solving the optimization problems like offsets, periods, and priorities assignment [16, 17, 35, 39].

Contribution. In our previous work [36], we proposed a method for the computation of the data chain worst-case latency of periodic independent tasks with implicit deadlines scheduled by a fixed-priority preemptive algorithm given a priority assignment of these tasks and their worst-case response times. As the method has exponential time complexity, in this work we:

- derive a polynomial-time upper bound on the data chain worst-case latency,
- identify the sources of bound pessimism and provide a condition for bound tightness,
- evaluate the impact of different worst-case response time estimations on the precision of the latency analysis for different chain lengths.

Paper structure. The paper is organized as follows. The data chain model is introduced in Section 2. We present a method for the calculation of the worst-case latency of the data chains in Section 3. As the method has an exponential time complexity, we propose in Section 4 a polynomial-time upper bound on the worst-case latency of the data chains. In Section 5, we evaluate the bound precision and efficiency using the generic task sets based on a real-world automotive system. We review the related work in Section 6 and conclude the paper in Section 7.

2 System model

We introduce a data chain as a sequence of independent periodic tasks with implicit deadlines scheduled by a fixed-priority preemptive policy upon a single processor.

2.1 Periodic task model

A system consists of a finite set of periodic tasks. Each task τ_i is characterized by its *worst-case execution time* C_i and its *period* T_i (both positive integers). An instance (job) of task τ_i is released every period T_i and must meet its execution requirement upper bounded by C_i before the arrival of its next instance (implicit deadline $D_i = T_i$). The first instance of τ_i is released at the time instant 0. The infinite set of all the release time instants of task τ_i is referred to as $A(\tau_i) = \{0, T_i, 2T_i, \ldots\}$. The $j-th$ instance of the task τ_i released at $r_{i,j} \in A(\tau_i)$ is denoted by $J_i(r_{i,j})$. In the rest of the paper, to facilitate the reading, an arrival $r_{i,j}$ of a task τ_i will be sometimes simply denoted by r_i when there is no possible ambiguity with different arrivals of τ_i. The *hyperperiod* H is the least common multiple of all task periods, and $T_{\max}$ denotes the largest period.

Tasks are scheduled upon a single processor by a fixed-priority preemptive scheduler. Each task τ_i is assumed to have a unique priority $\pi(\tau_i)$. We consider that τ_i has a higher priority than τ_k if $\pi(\tau_i) > \pi(\tau_k)$. Additionally, we denote a set of tasks with priorities higher than $\pi(\tau_i)$ as: $hp(\tau_i) = \{\tau_k : \pi(\tau_k) > \pi(\tau_i)\}$ and with priorities higher than or equal to $\pi(\tau_i)$ as: $hep(\tau_i) = hp(\tau_i) \cup \{\tau_i\}$. The tasks are assumed to be independent, and their executions cannot be blocked by another task other than due to contention for the processor. Once a task starts to execute, it will not voluntarily suspend its execution. Tasks can begin processing, complete, or be preempted at any time with respect to their parameters.

2.2 Task response time

The worst-case *job response time* $R_i(r_{i,j})$ of task τ_i released at $r_{i,j} \in A(\tau_i)$ is given by the longest time from $r_{i,j}$ until it completes its execution. The time instant when it completes its execution is denoted by $f_i(r_{i,j}) = r_{i,j} + R_i(r_{i,j})$.

The worst-case *task response time* of τ_i, denoted as R_i, is given by the maximum worst-case job response time among all its jobs: $R_i = \max_{r_{i,j} \in A(\tau_i)} R_i(r_{i,j})$. A system is deemed schedulable when $\forall \tau_i : R_i \leq T_i$.

2.3 Data chain model

A data chain F_n is a sequence of $n \geq 1$ tasks $(\tau_1, \ldots, \tau_n)$ describing a flow of communication between the tasks realized via shared registers. Let $head(F_n) = \tau_1$ be its first task, $last(F_n) = \tau_n$ its last task and $tail(F_n) = (\tau_2, \ldots, \tau_n)$ the chain obtained by removing τ_1 from F_n. If $n = 1$, then τ_1 after its start reads the data from its sensor, computes its outputs and writes the outputs to the appropriate actuator. Otherwise, if $n \geq 2$, the data chain F_n processes the data as follows:

- when τ_1 starts, it reads the data from a sensor, computes the results and, at the end of its execution, writes these results into the register of τ_2;
- when τ_i (for $i : 1 < i < n$) starts, it reads the data from the register of τ_{i-1}, computes the results and writes these results, at the end of τ_i, into the register of τ_{i+1};
- when τ_n starts, it reads the data from τ_{n-1}, computes the outputs and, at the end of its execution, writes these outputs to the corresponding actuator.

The start of the tasks does not depend on the communication, and the tasks are scheduled according to the assigned priorities only. Figure 2 illustrates a data chain F_3 of three tasks (τ_1, τ_2, and τ_3) and their registers.

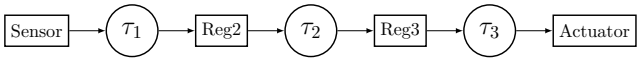


Figure 2: Register-based communication in a data chain of three tasks τ_1, τ_2, and τ_3.

The communication described above assumes the read-execute-write task semantics (implicit communication [6, 30] in AUTOSAR): the input data can only be read at the beginning of the task execution, and the results can only be written at the end of its execution. The same mechanism is implemented in Cyclical Asynchronous Buffers [15]. The operations on the registers (read and write) are atomic and are assumed to take a negligible amount of time. The computation process can be preempted at any time and resumed later from the same context. Each task has its individual register which stores only the most recent data. A data remains present in a register until being overwritten, and the access to the registers is asynchronous (i.e., can be accessed anytime).

2.4 Data propagation path

Based on the concept of timed path [21, 8], we define, for a data chain F_n released at time instant $r_1 \in A(\tau_1)$ such that $\tau_1 \in head(F_n)$, a set of *data propagation paths* $\Omega(F_n, r_1)$. A data propagation path $p \in \Omega(F_n, r_1)$ is an n-tuple $(r_1, \ldots, r_n)$ where $r_i \in p$ is the release time of the instance of task τ_i that propagates the data. A task τ_i propagates a data when it writes this data for the first time into a task register of the next task τ_{i+1}. It is supposed that from the time instant r_1 the data is continuously present at the chain's input, and the registers of the other tasks are empty at r_1.

Figure 3 shows two different execution scenarios of a data chain F_3. The chain is composed of three tasks: τ_1, τ_2, and τ_3 such that $\pi(\tau_2) > \pi(\tau_3) > \pi(\tau_1)$. Two data propagation paths can be distinguished for $\Omega(F_3, r_{1,1})$. The first one, shown in Figure 3 a), occurs when τ_1 terminates at $f_{1,1}$: $(r_{1,1}, r_{2,3}, r_{3,3})$. The second one, shown in Figure 3 b), occurs when τ_1 terminates earlier at $f'_{1,1}$: $(r_{1,1}, r_{2,2}, r_{3,2})$ with $f_{1,1} > f'_{1,1}$.

2.5 Data chain latency

The latency of a data chain is a time duration between the arrival of its input and the first output corresponding to this arrival [56].

Definition 1 (Latency)**.** *Let F_n be a data chain whose registers are initially empty. Its data latency is a time interval elapsed between:*

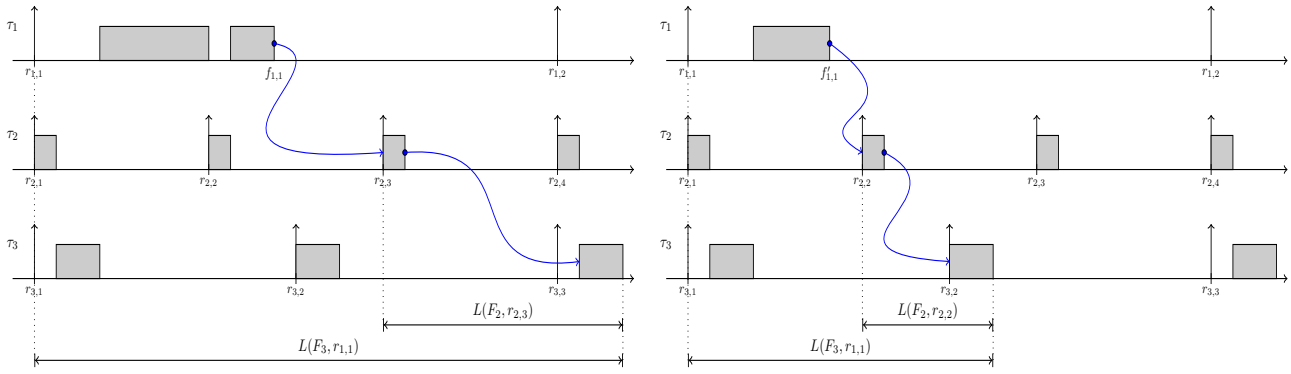
- *the time instant at which the data arrives at the input sensor connected to $head(F_n)$ given that the data is available sufficiently long to be detected, and*
- *the time instant at which the actuator connected to $last(F_n)$ is updated with the data corresponding to the computation performed by all the tasks of F_n.*

Given that the data arrives at r_1, the maximum latency $L(F_n, r_1)$ of a data chain F_n corresponds to the data propagation path in which the last task τ_n completes at the latest time.

Definition 2 (Maximum data chain latency at r_1)**.** *Let F_n be a data chain of $n \geq 1$ tasks and $r_1 \in A(\tau_1)$ the release time of its first task $\tau_1 = head(F_n)$ such that the input of τ_1 arrives in its register at r_1. The maximum latency of data chain F_n released at r_1 is:*

$$L(F_n, r_1) = \max_{p \in \Omega(F_n, r_1)} f_n(r_n) - r_1 \qquad (1)$$

where $r_n \in p$ and $\tau_n = last(F_n)$.



(a) Task τ_1 finishes at $f_{1,1}$.

(b) Task τ_1 finishes at $f'_{1,1}$.

Figure 3: Two possible execution scenarios of a data chain of three tasks: τ_1, τ_2, and τ_3.

3 Latency analysis

In this section, we present a method for computing the maximum latency of the data chains of real-time periodic tasks [36]. First, we characterize the time durations between the releases of a task that writes the data and the next task in the chain that reads this data. Given a data arrival time, we construct a sequence of releases of the tasks propagating the data from the first to the last task in the chain (data propagation path). Then, to obtain the worst-case latency, we examine all possible data arrival time instants over the hyperperiod.

3.1 The latest data propagation time

We characterize the longest delay in data passing for a pair of directly communicating tasks. Given a release time of the task propagating data (producer), we find the latest possible release time instant of the task that reads this data for the first time (consumer).

In the example shown in Figure 3, there are two instances of task τ_2 that can propagate the data. Figure 3 a) shows the inter-task data passing when τ_1 finishes at $f_{1,1}$ (suppose that $f_1(r_{1,1}) = f_{1,1}$) and the data is consumed by τ_2 released at $r_{2,3}$. Figure 3 b) corresponds to the case when τ_1 finishes at $f'_{1,1}$ ($f'_{1,1} < f_{1,1}$) and the data is consumed by τ_2 released at $r_{2,2}$ ($r_{2,2} < r_{2,3}$). Task τ_2 instance that consumes the data propagates it further to τ_3. In this example, the data from τ_1 can be propagated to τ_3 by the instances of τ_2 released at $r_{2,2}$ or $r_{2,3}$. The time instant $r_{2,3}$ is the latest possible release of τ_2 that can propagate the data.

Let F_2 be a data chain of two communicating tasks: producer τ_p and consumer τ_c having respectively periods T_p and T_c. The producer, at the end of its execution, fills the input register of the consumer with the result of its computation. Let r_c be the latest release time of the consumer job that has for the first time in its register the data written by the producer released at r_p.

First, suppose that the consumer has a higher priority than the producer: $\pi(\tau_c) > \pi(\tau_p)$. Figure 4 illustrates a sample execution of such two tasks. Due to the priority order, consumer τ_c can preempt producer τ_p. The consumer checks for the data at each release but the data cannot be written into the register before the end of the producer execution. The consumer instance released immediately after the end of the producer reads the data

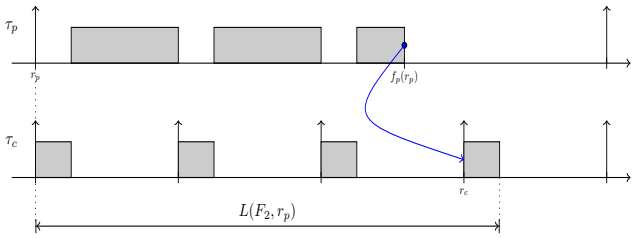


Figure 4: Communicating tasks: consumer τ_c has a higher priority than producer τ_p.

for the first time. At worst, the producer finishes at $f_p(r_p)$, and the data is read by the earliest consumer instance released after $f_p(r_p)$:

$$r_c = \left\lceil \frac{f_p(r_p)}{T_c} \right\rceil T_c \tag{2}$$

Now, suppose that the producer has a higher priority than the consumer: $\pi(\tau_p) > \pi(\tau_c)$. Suppose also that the task set contains a third task τ_h which is not part of the chain. Task τ_h has the highest priority, $\pi(\tau_h) > \pi(\tau_p) > \pi(\tau_c)$, and its period is equal to the consumer period, $T_h = T_c$. A sample execution of such three tasks is shown in Figure 5. The consumer has a lower priority

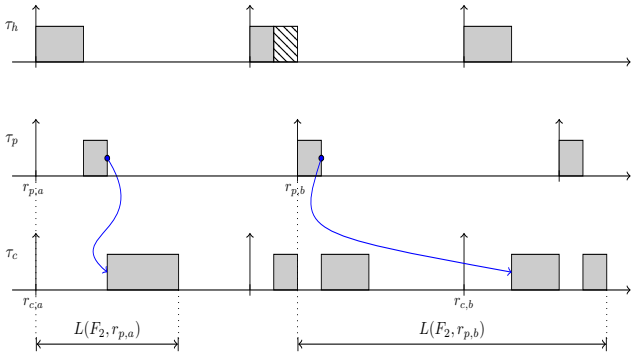


Figure 5: Communicating tasks: producer τ_p has a higher priority than consumer τ_c.

and cannot preempt the producer. Suppose that the producer and the consumer are:

a) released at the same time (see time instant $r_{p,a}$ in Figure 5): Due to the priority order, the consumer

waits until the producer completes its execution. Then, it gets the data without any delay when starting its execution.

b) not released at the same time (see time instant $r_{p,b}$ in Figure 5): If not blocked by the higher priority jobs (e.g., by the second instance of task τ_h having an execution time shorter than its worst-case value, the difference is illustrated by the dashed area), the consumer starts before the current producer instance and reads the data written in its register by the previous producer instance. The data written by the current producer instance will be made available for the next consumer instance.

At worst, the data is read for the first time by the first consumer instance released at or after the release of producer:

$$r_c = \left\lceil \frac{r_p}{T_c} \right\rceil T_c \tag{3}$$

Lemma 1. *In a schedulable task set with two communicating tasks τ_p and τ_c such that τ_p writes the results of its computation to the register of τ_c, the latest possible release r_c of the instance of τ_c reading for the first time the data propagated by the instance of τ_p released at $r_p \in A(\tau_p)$ is given by:*

$$r_c = \begin{cases} \left\lceil \frac{f_p(r_p)}{T_c} \right\rceil T_c & \text{if } \pi(\tau_p) < \pi(\tau_c) \\ \left\lceil \frac{r_p}{T_c} \right\rceil T_c & \text{if } \pi(\tau_p) > \pi(\tau_c) \end{cases} \tag{4}$$

where T_p and T_c are periods of τ_p and τ_c, respectively.

3.2 The longest data propagation path

We characterize the data propagation path that gives a maximum latency value.

A data propagation path in which the data propagation occurs for each task as late as possible and each task can attain its worst-case job response times, leads to the maximum latency. For example, the data path shown in Figure 3 a) satisfies the conditions mentioned above. The value of its latency is maximal. Propagating the data earlier, as for the data path from Figure 3 b), cannot increase the latency.

If such a path cannot be constructed in the schedule, then its latency value cannot be lower than the actual value of the latency given by some realizable path.

Lemma 2. *For any data chain of $n \geq 1$ schedulable tasks, the data propagation path p in which each task instance that propagates the data is released as late as possible, and each task instance can terminate at its worst-case finishing time, gives the maximum latency.*

The proof of Lemma 2 is reported in Appendix.

3.3 Method for latency computation

We compute the maximum latency value for a data chain of $n \geq 1$ tasks by finding the latest possible releases of the tasks that consume the data.

Consider a data chain of n tasks released at r_1. The first task of the data chain is a producer, and the second task is a consumer. We find the latest release time instant of the consumer using Equation (4). Then, the second task becomes the new producer, and the third task the new consumer. The release of the producer is the release of the previous consumer, and the same reasoning as for the first two tasks is applied. The method continues in like manner for every successive pair of the chain tasks and can be implemented in a recursive manner.

Consider again Figure 3 and data chain F_3 with τ_1 released at $r_{1,1}$ and completed at $f_{1,1}$. The maximum latency $L(F_3, r_{1,1})$ can be expressed as $L(F_3, r_{1,1}) = r_{2,3} + L(F_2, r_{2,3}) - r_{1,1}$ where F_2 is a data chain composed of τ_2 and τ_3.

Theorem 1. *Let F_n be a data chain of $n \geq 2$ tasks and F_{n-1} be a data chain of $n-1$ tasks obtained from F_n by removing its first task τ_p: $F_{n-1} = tail(F_n)$. The maximum latency $L(F_n, r_p)$ of F_n when its first task τ_p is released at the time instant $r_p \in A(\tau_p)$ is:*

$$L(F_n, r_p) \;\geq\; (r_c - r_p) + L(F_{n-1}, r_c) \tag{5}$$

where $L(F_{n-1}, r_c)$ is the maximum latency of F_{n-1} whose first task τ_c is released at the time instant r_c given by Formula (4).

Proof. Let $(r_1, \ldots, r_n)$ be a data propagation path obtained by the iterative application of Theorem 1. For $1 < i \leq n$, Equation (4) gives r_i that is the latest possible release of τ_i instance that propagates the data. By Lemma 2, a latency value of a data propagation path with the latest possible release time instants cannot be smaller than a latency value of any other data propagation path. □

Algorithm 1 uses the results of Theorem 1 to compute the maximum latency of a data chain F_n whose first task is released at the time instant r_p. The algorithm has a linear time complexity $O(n)$ (we suppose that the worst-case response times are known).

Algorithm 1 Data chain maximum latency at r_p.

Require: F_n is a data chain, $n \geq 1$, $r_p \in A(head(F_n))$

```
1:  function L(F_n, r_p)
2:      τ_p ← head(F_n)
3:      if n = 1 then
4:          return R_p(r_p)
5:      end if
6:      F_{n-1} ← tail(F_n), τ_c ← head(F_{n-1}), Q ← 0
7:      if π(τ_p) < π(τ_c) then
8:          Q ← R_p(r_p)
9:      end if
10:     r_c ← ⌈(r_p + Q)/T_c⌉ T_c
11:     return r_c − r_p + L(F_{n-1}, r_c)
12: end function
```

3.4 The worst-case latency

The worst-case latency can be calculated by applying the above-presented analysis for all the release time instants of the first task in the chain and by taking into account the delay of the first task sensor data detection.

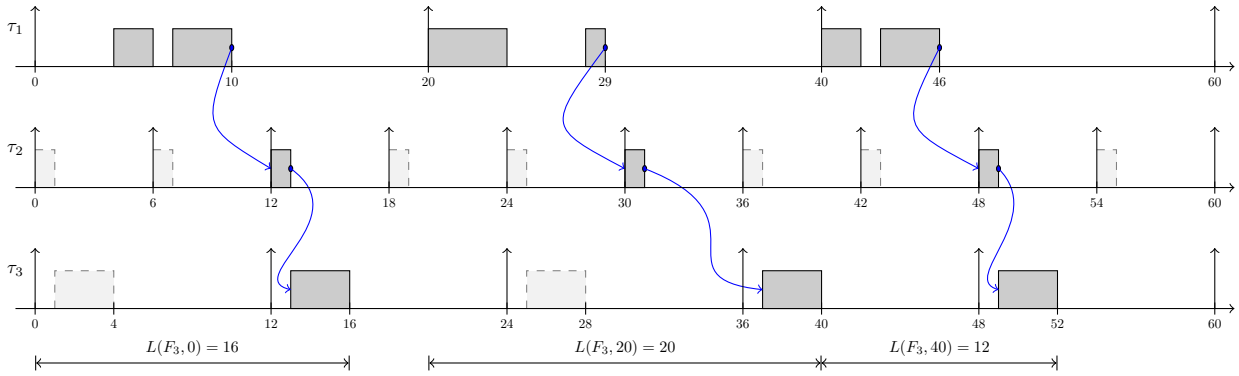


Figure 6: Maximum latency computation for the data chain F_3 of three tasks τ_1, τ_2 and τ_3 with the following parameters: ($C_1 = 5$, $T_1 = 20$, $\pi(\tau_1) = 1$), ($C_2 = 1$, $T_2 = 6$, $\pi(\tau_2) = 3$) and ($C_3 = 3$, $T_3 = 12$, $\pi(\tau_3) = 2$).

At the earliest, a task can start to execute at its release time (if the higher priority jobs are completed). Suppose that τ_1, the first task in the chain, starts at $r_1 \in A(\tau_1)$, and the data arrives at the sensor immediately after its start. In that case, the data is handled by the next τ_1 instance released at $r_1 + T_1$. The delay experienced by the data in the sensor register is not greater than one task period (similar reasoning is used in [16, 30]). Then, the data travels along the data propagation path, as explained above. As there are many paths and their trajectories depend on the release time of the first task, to find the worst-case latency value all these paths must be analyzed. Given a data chain F_n of $n \geq 1$ tasks, the value of its worst-case latency is:

$$L(F_n) < T_1 + \max_{r_1 \in A(\tau_1)} L(F_n, r_1) \tag{6}$$

where $\tau_1 \in head(F_n)$ and T_1 is the period of τ_1.

To compute the worst-case latency, Formula (6) must be evaluated for all releases of the first chain task $(0, T_1, 2T_1, \ldots)$. Using the fact that the schedule repeats cyclically every hyperperiod H [40], it is sufficient to check the releases of the first chain task within the interval $[0, H)$. Moreover, since the execution of the task is not interfered by the lower priority tasks, it is sufficient to examine only the releases of the first chain task within the interval $[0, H)$ where $H = \text{lcm}\{\, T_j \mid \tau_j \in hep(\tau_i), \tau_i \in F_n \,\}$. Depending on system properties, this limit can be further reduced. If the worst-case response times of all task instances are equal, $\forall k \in \mathbb{N}, \tau_i : R_i(t) = R_i(t + kT_i)$, then it is sufficient to apply Algorithm 1 on the hyperperiod of all the tasks in the data chain $H = \text{lcm}\{\, T_i \mid \tau_i \in F_n \,\}$. For instance, in the analysis of task sets with *harmonic* periods scheduled under *Rate Monotonic* (*RM*) all the jobs of the same task have their worst-case job response times equal [43, 65]. Such a hypothesis can also be made when the schedulability analysis provides only a single upper bound for all task jobs.

Algorithm 2 finds a maximum latency by calling the function $L(F_n, r_1)$, defined in Algorithm 1, for all the possible releases r_1 of the first task in the chain F_n within the hyperperiod H. Then, finding the worst-case latency has an $O(n \cdot \frac{H}{T_1})$ time complexity. The input size of the problem is bounded by $O(n \cdot \log(T_{\max}))$. Since the hyperperiod grows exponentially as a function of the largest task period in the task set and with the number of tasks [28], finding the worst-case latency has an *exponential* time complexity and may be intractable. Nonetheless, in the presence of harmonic periods, the hyperperiod H is equal to the largest task period $T_{\max}$. Thus, finding the worst-case runs in polynomial time with respect to the numeric value of the input and has a *pseudo-polynomial* time complexity.

Algorithm 2 Data chain worst-case latency.

Require: F_n is a data chain of n tasks, $n \geq 1$
1: $H \leftarrow$ hyperperiod of all tasks in the set, $\tau_1 \leftarrow head(F_n)$
2: **for all** $r_1 \in \{0, T_1, 2T_1, \ldots, H - T_1\}$ **do**
3: compute $L(F_n, r_1)$ using Algorithm 1
4: **end for**
5: $L(F_n) \leftarrow T_1 + \max_{r_1} L(F_n, r_1)$

3.5 Example

Figure 6 illustrates the worst-case latency computation for a sample execution of a data chain F_3 composed of three tasks: τ_1, τ_2 and τ_3. The task instances that do not propagate new data are shadowed.

For the instances of task τ_1 released at time instants 0, 20 and 40, the maximum latency values are respectively $L(F_3, 0) = 16$, $L(F_3, 20) = 20$ and $L(F_3, 40) = 12$. By adding the delay of the first task sensor data detection equal to its period ($T_1 = 20$) to the greatest maximum latency value ($L(F_3, 20) = 20$) we obtain the worst-case latency $L(F_3) = 40$.

In this example, we always consider the worst-case job response times. If the worst-case task response times were used instead, the worst-case latency would be associated with the data path released at time instant 40 and would be equal to 44 overestimating the actual value by 10% (since $R(\tau_1) = 10$, it would be considered that $J_1(40)$ propagates the data to $J_2(54)$). In Section 5, we empirically evaluate the impact of the precision in the response-time analysis on the latency over-approximation.

In the data path leading to the worst-case latency, the delay in the data passing between τ_1 and τ_2 is lower than in the data path released at 0. In this schedule, there is no data path in which every pair of communicating tasks always incurs the delay that is greater than in

all other data paths. Nonetheless, in the next section, we will suppose that every pair of communicating tasks can always cause the greatest possible delay to obtain a latency upper bound that can be computed in polynomial time in the input size.

4 Latency bound

We derive an upper bound on the worst-case latency of the data chain. This bound can be computed in polynomial-time in the input size by finding the largest time intervals between the releases of any two directly communicating tasks.

4.1 Bound derivation

We characterize a relation between the releases of two communicating real-time periodic tasks. Let τ_i and τ_j be two periodic tasks such that τ_i writes into τ_j register. If T_i is the period of τ_i and T_j is the period of τ_j, then their releases are given by $r_i = k_i T_i$ and $r_j = k_j T_j$ for any $k_i, k_j \in \mathbb{N}_0$. Based on the Bézout's identity and its implications, $r_j - r_i = k_j T_j - k_i T_i$ must be multiple of $\gcd(T_i, T_j)$ and every multiple of $\gcd(T_i, T_j)$ can correspond to some distance between the releases of these tasks (the same reasoning is used in [51, 52, 53]). Consequently, in a schedule of periodic tasks without offsets, for any two tasks τ_i and τ_j, the following condition must be verified:

$$\forall\, (r_i, r_j) \in A(\tau_i) \times A(\tau_j) :\ \gcd(T_i, T_j) \,|\, r_j - r_i \qquad (7)$$

Consider first the case when the producer has a higher priority than the consumer: $\pi(\tau_p) > \pi(\tau_c)$. Lemma 1 states that the data, at worst, must be processed by the first job of consumer released at or after the current producer release. As the consumer is released every T_c, one of its jobs is always released within the interval $[r_p, r_p + T_c)$.

$$if\ \pi(\tau_p) > \pi(\tau_c) : \quad r_c - r_p \ < \ T_c \qquad (8)$$

Now we consider the case when the consumer has a higher priority than the producer: $\pi(\tau_c) > \pi(\tau_p)$. Lemma 1 shows that the data must be processed by the first consumer job released immediately after the end of the producer. The producer cannot take more time than R_p to produce the data. Additionally, like in the previous case, the consumer needs less than T_c units to get the data once it is written in its register.

$$if\ \pi(\tau_c) > \pi(\tau_p) : \quad r_c - r_p \ < \ R_p + T_c \qquad (9)$$

The length of the time interval between r_p and r_c can be bounded by Formulas (8) and (9). By the condition given in Formula (7), the length of this interval must be multiple of $\gcd(T_p, T_c)$. Based on these properties, we can obtain an upper bound on the time interval between r_p and r_c.

Lemma 3. *Let τ_p and τ_c be two communicating real-time periodic tasks such that τ_p writes the results of its computation to the register of τ_c. The maximal distance between the release time r_p of the task τ_p and the release time r_c of the task τ_c that reads the data produced by τ_p at r_p for the first time is not greater than:*

$$r_c - r_p \ \leq \ \left(\left\lceil \frac{q(\tau_p, \tau_c) \cdot R_p}{\gcd(T_p, T_c)} \right\rceil - 1 \right) \gcd(T_p, T_c) + T_c \qquad (10)$$

where

$$q(\tau_p, \tau_c) = \begin{cases} 1 \ \textit{if}\ \pi(\tau_c) > \pi(\tau_p)\ , \\ 0 \ \textit{if}\ \pi(\tau_c) < \pi(\tau_p). \end{cases}$$

Proof. In order to ease the subsequent presentation, we introduce the variable $X = q(\tau_p, \tau_c) \cdot R_p(r_p)$. Equation (4) can be rewritten as:

$$r_c = \left\lceil \frac{r_p + X}{T_c} \right\rceil T_c$$

The following relation must be satisfied:

$$\left\lceil \frac{r_p + X}{T_c} \right\rceil T_c \ < \ \left\lfloor \frac{r_p + X + T_c}{T_c} \right\rfloor T_c \ \leq \ r_p + X + T_c$$

The value of $r_c - r_p$ can be therefore bounded by:

$$r_c - r_p = \left\lceil \frac{r_p + X}{T_c} \right\rceil T_c - r_p \ < \ X + T_c \qquad (11)$$

Time instant r_p is a multiple of T_p, and time instant r_c is a multiple of T_c. Their greatest common divisor $\gcd(T_p, T_c)$ divides both instants, and thus there must exist $k, k' \in \mathbb{N}_0$ such that $r_p = k \gcd(T_p, T_c)$ and $r_c = k' \gcd(T_p, T_c)$. Consequently, $r_c - r_p = (k' - k) \gcd(T_p, T_c)$ is also a multiple of $\gcd(T_p, T_c)$. The greatest multiple of $\gcd(T_p, T_c)$ satisfying the right-hand side of Expression (11) can be used as a bound for $r_c - r_p$.

$$r_c - r_p \ \leq \ \left(\left\lceil \frac{X + T_c}{\gcd(T_p, T_c)} \right\rceil - 1 \right) \gcd(T_p, T_c)$$

T_c is divisible by $\gcd(T_p, T_c)$.

$$r_c - r_p \ \leq \ \left(\left\lceil \frac{X}{\gcd(T_p, T_c)} \right\rceil - 1 \right) \gcd(T_p, T_c) + T_c$$

Finally, if $\pi(\tau_p) < \pi(\tau_c)$ then $X = q(\tau_p, \tau_c) \cdot R_p(r_p) = 0$ and we get Formula (10). Otherwise, if $\pi(\tau_p) > \pi(\tau_c)$ then $X = q(\tau_p, \tau_c) \cdot R_p(r_p) \leq q(\tau_p, \tau_c) \cdot R_p$ and we get Formula (10) as well. □

4.2 Bound computation and its time complexity

Algorithm 3 computes the upper bound on the worst-case latency of a data chain.

Its running time is linear in the number of tasks in the chain. The greatest common divisor is calculated with the Euclidean algorithm. According to Lamé's Theorem, the number of steps in the Euclidean algorithm never exceeds five times the number of digits in the smaller number [32, 37]. If $T_{\max}$ is the maximal period among all periods $T_1, \ldots, T_n$, then the worst-case complexity of the Euclidean algorithm applied to any pair of these periods is not greater than $5 \cdot \lceil \log_{10}(T_{\max} + 1) \rceil$. Thus, the overall running time of Algorithm 3 is bounded by $O(n \cdot 5 \cdot \lceil \log_{10}(T_{\max} + 1) \rceil)$ which is polynomially bounded

Algorithm 3 Data chain worst-case latency upper bound.

Require: F_n is a data chain, $n \geq 1$
Ensure: $\hat{L} \geq L(F_n)$
1: $\hat{L} \leftarrow T_1$
2: **for** $i \leftarrow 1, n-1$ **do**
3: $\quad \tau_p \leftarrow \tau_i,\ \tau_c \leftarrow \tau_{i+1}$
4: $\quad \hat{L} \leftarrow \hat{L} + T_c - \gcd(T_p, T_c)$
5: $\quad$ **if** $\pi(\tau_c) > \pi(\tau_p)$ **then**
6: $\quad\quad \hat{L} \leftarrow \hat{L} + \left\lceil \frac{R_p}{\gcd(T_p, T_c)} \right\rceil \gcd(T_p, T_c)$
7: $\quad$ **end if**
8: **end for**
9: $\hat{L} \leftarrow \hat{L} + R_n$

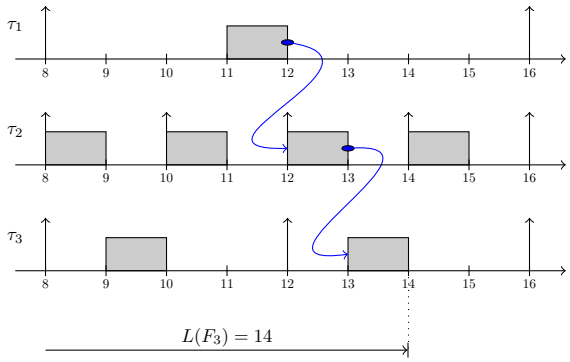


Figure 7: Latency bound overestimation for a data chain of three tasks with harmonic periods under RM.

in the input length. Note that the algorithm complexity is independent of the response time analysis as it assumes that the worst-case response times are known. Indeed, the algorithm can be combined with the response time analyses characterized with different complexities and with different precisions. The experiments performed in the following section try to find a trade-off between complexity and accuracy of the response time analysis.

4.3 Bound tightness

The bound can overestimate the value of the worst-case latency given by the method described in the previous section. In the following, we review several causes of this pessimism.

First, the bound considers that every task instance can reach the worst-case task response time: $\forall \tau_i\, \forall t \geq 0 : R_i(t) = R_i$. Second, the bound may consider schedules than cannot exist: Formula (10) seeks the maximum distance between the releases of two consecutive tasks within the chain, but it does not check if this distance is in accordance with other releases of the tasks in this chain. More precisely, given any two tasks in the chain τ_i and τ_j, the condition given by Formula (7) is verified only for i and $j = i + 1$ but in the valid schedule it must be verified for all other $j : 1 \leq j \leq n$ (e.g., when computing the bound and $n \geq 3$, then the relation is never checked between the releases of τ_1 and τ_n).

The bound from Lemma 3 gives a time interval between the releases of two consecutive tasks of a data chain. Let $\Delta_{k,k+1} \geq 0$ be such a time interval between the releases of tasks τ_k and τ_{k+1} for $k \geq 1$. Therefore, the time interval between the releases of any two tasks τ_i and τ_j of this chain, such that $j > i \geq 1$, is given by $\Delta_{i,j} = \sum_{k=i}^{j-1} \Delta_{k,k+1}$. All the values calculated in this way, for all $i, j \leq n$, must satisfy Condition (7) if the results of applying bound from Lemma 3 correspond to a schedule that can be actually observed. Otherwise, the results are overestimated by considering a schedule that cannot exist.

Lemma 3, for tasks τ_k and τ_{k+1} within a data chain, gives a time interval that is a multiple of $\gcd(T_k, T_{k+1})$ (see Equation (10)). The time interval between the releases of τ_i and τ_j, calculated as it is explained above, can be expressed as $\Delta_{i,j} = \sum_{k=i}^{j-1} x_k \gcd(T_k, T_{k+1})$ where $x_k \in \mathbb{N}_0$. From the properties of the greatest common divisor, it can be shown that $\Delta_{i,j} = x_i \gcd(T_i, T_{i+1}) + \ldots + x_{j-1} \gcd(T_{j-1}, T_j)$ is a multiple of $\gcd(T_i, \ldots, T_j)$. Now, suppose that the following relation holds for all the tasks:

$$\forall i, j \leq n : \gcd(T_i, T_j) = \gcd(T_i, \ldots, T_j) \tag{12}$$

then the Condition (7) is satisfied, and it can be concluded that Lemma 3 yields the time intervals from which a valid schedule can always be constructed.

The condition given in Formula (12) is not always satisfied by harmonic periods. Indeed, consider the worst-case latency of the data chain shown in Figure 7. The chain is composed of three tasks τ_1, τ_2, and τ_3 with harmonic periods and priorities assigned according to *RM* scheduling policy. The task set parameters are described in Table 1. The Condition (12) is not satisfied in that case as $\gcd(T_1, T_3) \neq \gcd(T_1, T_2, T_3)$ (i.e. $4 \neq 2$).

Table 1: Task parameters for the example from Figure 7.

Task	T_i	C_i	$\pi(\tau_i)$
τ_1	8	1	1
τ_2	2	1	3
τ_3	4	1	2

We assume that the input data arrives right after the time instant 0 (this part of the schedule is skipped in Figure 7). The actual value of the worst-case latency is 14. The value of the worst-case latency calculated with Algorithm 3 is equal to 16. The algorithm overestimates the actual value of latency when considering the data passing between the tasks τ_2 and τ_3. It assumes the worst-case delay is equal to 2 while, in reality, this delay is always 0.

Under certain assumptions, the bound can be less pessimistic or can even give the same value of the worst-case latency as the method described in Section 3. If the chain is composed of two tasks only, τ_1 and τ_2, then the distance between the task releases given by Formula (10) corresponds always to a valid schedule: the distance is multiple of $\gcd(T_1, T_2)$ and this is the only condition required by Formula (7). The same follows if all the task periods in the chain are equal: $\forall i, j \leq n : T_i = T_j$. Another source of pessimism comes from considering for each task instance its worst-case task response time as its

worst-case job response time. However, as mentioned in the previous section, under certain conditions, like in the harmonic schedules under *RM*, this is actually the case.

5 Experiments

In this section, we evaluate the performance of the worst-case latency analysis [36] summarized in Section 3 and of the polynomial-time upper bound on the latency presented in Section 4 in terms of precision and practical complexity on a generic automotive benchmark [38] upon a single processor.

We compare these two methods against a linear-time upper bound on the worst-case latency proposed by Davare et al. [16]. In principle, the latter method applies for sporadic tasks, but its use may also be advantageous for the periodic tasks due to its linear time-complexity. As the tasks are uncoordinated, the producer τ_p may write the data into the register of the consumer task τ_c immediately after its start. It induces a maximum delay of $T_c + R_c$ for each pair of producer/consumer. The worst-case latency for a chain of n tasks F_n is expressed as: $L(F_n) = \sum_{i=1}^{n} (T_i + R_i)$.

The algorithms described in this work are available on-line [2] through a Python implementation with which experiments presented below are fully reproducible.

5.1 Chain generation

Data chains are made of tasks picked from a set of 50 tasks. Task parameters are generated based on the automotive benchmark [38]. Periods are randomly chosen from the set $\{1, 2, 5, 10, 20, 50, 100, 200, 1000\}$ with a probability of appearance taken from [38]. The utilization of task τ_i is generated using the UUnifast [11] algorithm and then multiplied by the chosen period T_i giving C_i. We consider RM scheduling policy and discard unschedulable task sets. To focus on chain length only for Figures 8a, 8b and 8c, we successively decrease the chain length in such a way that the number of distinct periods remains respected.

5.2 Analysis and bounds precision

We evaluate the precision of the analysis and bound respectively described in Section 3 and 4 as well as the bound of Davare et al. [16]. They require worst-case response time (WCRT) information. In Figure 8, we compare the results varying the precision of the method used to obtain WCRT. We study the impacts of the chain length and of the number of distinct periods in the chain on the latency. For each task instance of τ_i released at $r_{i,j} \in A(\tau_i)$, we use the following methods to safely estimate its job WCRT $R_i(r_{i,j})$:

- SimSo simulator [14] for its exact value;
- exact schedulability test [33] for R_i and then $R_i(r_{i,j}) = R_i$ (R_i, circles);
- sufficient schedulability test [2] for an upper bound on R_i and thus it is considered that $R_i(r_{i,j})$ is equal to this bound (sufficient R_i, squares);
- no response time information, $R_i(r_{i,j})$ is bounded by T_i ($R_i = T_i$, triangles).

In Figures 8a, 8b and 8c, results are averaged for a given task set utilization factor u and a chain length. Utilization factors are selected from the set $\{0.25, 0.5, 0.75\}$ and the number of distinct periods from 1 to 5. In Figure 8d, results are averaged for a given number of distinct periods p, and u is set to 0.75. Each experiment was repeated 10000 times. In all the plots of Figure 8, the latency computed using Algorithm 1 corresponds to a precision ratio of 100% on the y-axis. A precision ratio is equal to:

$$\frac{L \text{ with estimated WCRT}}{L \text{ with Algorithm 2}} \cdot 100\% \tag{13}$$

In our experiments, the worst-case latency calculated with the worst-case job response times $R_i(r_{i,j})$ is equal to the worst-case latency calculated with the worst-case task response times R_i with a tolerance of 10^{-3} (and thus not represented in Figure 8a). Indeed, as task periods are quasi-harmonic, most of the task instances verify $R_i(r_{i,j}) = R_i$ under RM scheduling (see also Section 3.4).

We record the following observations:

- the proposed polynomial-time upper bound on latency, represented in Figure 8b (based on worst-case task response time R_i), has a maximum over-estimation of 10% of the worst-case latency computed with the exponential method presented in Section 3; considering the gap on the theoretical complexity, the pessimism introduced is very limited;
- to obtain the polynomial-time complexity of both, the response time and the latency analysis, the proposed polynomial-time upper bound can be combined with the utilization based schedulability tests [12, 41]; the over-estimation of the worst-case latency is then about 80% of the worst-case latency computed with the exponential method ($R_i = T_i$).
- for a similar theoretical complexity, the upper bound we propose (based on R_i) is more accurate than the method of Davare et al. of Figure 8c;
- the chain length has the effect of decreasing the pessimism of the worst-case job response time approximation (Figure 8a) while it remains quite stable for the bound (Figure 8b) and slightly increases for the method of Davare et al. (Figure 8c);
- the precision of the upper bound on latency decreases when the number of distinct periods increases (Figure 8d) because the bound is sensitive to the gcd of the tasks periods; as pointed out in Section 4, the upper bound on R_i is exact with equal periods in the chain;
- globally, the more precise the worst-case job response time approximation, the better precision of the bound.

[2] `www.lias-lab.fr/~antoinebertout/software/latency.zip`

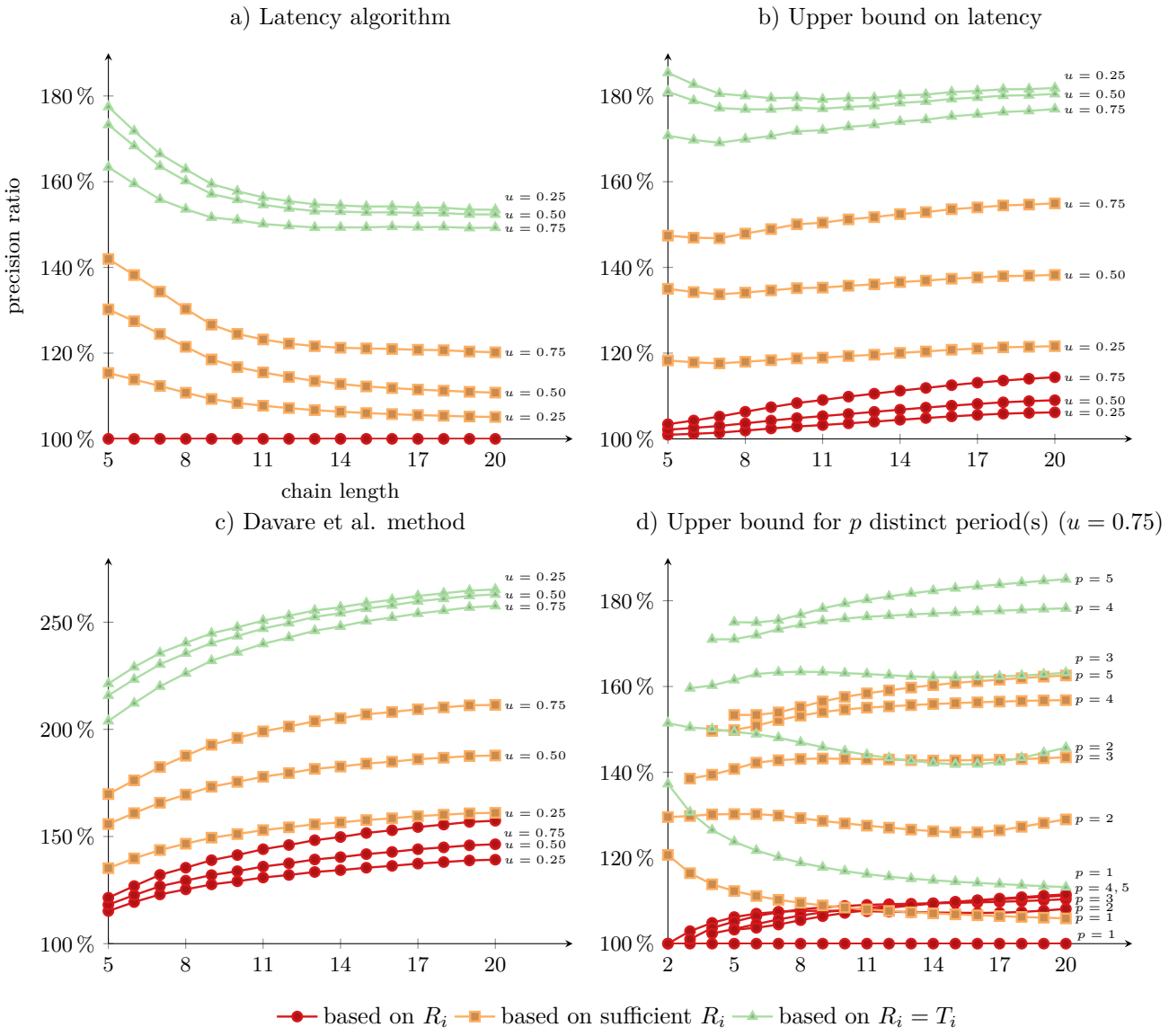


Figure 8: Impact of different WCRT estimations on the latency computations precision.

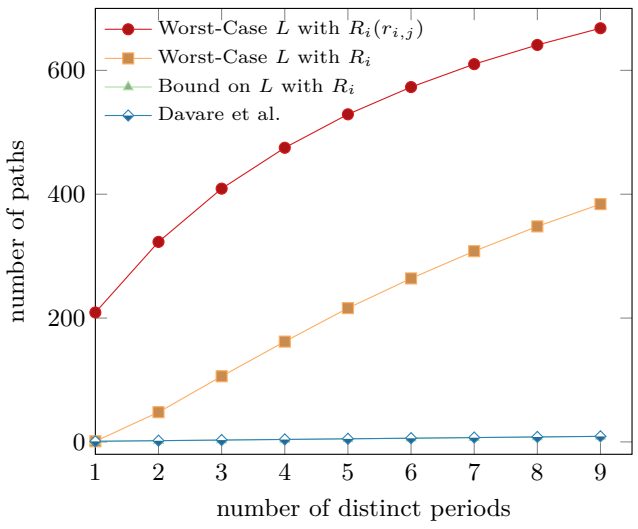


Figure 9: Average number of paths explored by the different latency computation methods.

5.3 Computational complexity

As explained in Section 3.4, the exact method evaluates one data propagation path per each release of the first task in the chain in H and has an $O(n \cdot \frac{H}{T_1})$ time complexity. We also proposed in Section 3.4 to use as a limit the hyperperiod of the chain tasks $H(F_n)$ if the worst-case task response times only are available.

Figure 9 shows the average number of data propagation paths traversed by each method as a function of the number of distinct periods in the chain. For each number of the distinct periods, the chains with every possible period combination (with replacement) were considered. As expected, we observe that our upper bound and the method of Davare (lines are overlaid) are the most efficient. The worst-case latency computation with the method described in Algorithm 2 must process, respectively, more than 300 and more than 600 data propagation paths with the worst-case task response times and with the worst-case job response times.

6 Related work

We survey related work on the latency computation for the periodic real-time task and similar models.

Harmonic periods. Gerber et al. [26, 27] consider a task chain with harmonicity constraint (i.e., periods that pairwise divide each other [31]) and enforce by the offset and priority assignment that the producer always executes before the consumer. Di Natale et al. [47] and Davare et al. [16] specify the general case when producer and consumer have harmonic periods without being necessarily monotonic in their lengths over the entire chain. They assume additionally that the relative offsets are selected to enforce the execution of the producer before the consumer. In the present work, there are no particular constraints neither on the task periods nor on the relative offsets.

Uncoordinated tasks. Davare et al. [16] also consider periodic tasks. The proposed method for the worst-case latency computation has a linear-time complexity. However, as the tasks are running on different nodes without a consistent view of time (no clock synchronization), the assumed worst-case behavior is more pessimistic than in the case of periodic tasks running on a single processor as supposed in this paper.

Periodic task model. Feiertag et al. [21] and Mubeen et al. [45] develop frameworks for the computation of different types of latencies under the same model as in the present work. In [5] Becker et al. propose a method for the worst-case latency computation using the information about the worst-case response times of the tasks. The authors develop an effective pruning technique to reduce the number of analyzed points in an inter-task communication tree. Following on from these works, we use the scheduling information (priority, response time) to identify directly with a closed-form expression the communication time instant leading to the maximum latency. None of these algorithms runs faster than exponential time. In this paper, we tackle their computational complexity by proposing an upper bound that can be computed in a polynomial-time. Rajeev et al. [56] and Anwikar and Badhuri [1] consider distributed systems in which chains are composed of preemptive tasks and non-preemptive messages. The systems are represented by a set of automata, and then the latencies are obtained using a model-checking based technique. Khatib et al. [34] use Synchronous Data-flow Graphs to model the communications between multi-periodic tasks. It is assumed that each task instance starts exactly at its release. A method for the latency computation, as well as a lower and an upper bound on its value, are proposed. Our model is based on the real-time periodic task whose start does not necessarily coincide with its release.

Abstract models. Wyss et al. [63, 64] present an analysis of the end-to-end latencies for a formal language with synchronous semantics named Prelude [23, 50]. The data propagation in a functional chain is expressed by "a data dependency word" [24, 50]. Biondi et al. [13] and Martinez et al. [42] address the problem of end-to-end latency for the tasks based on the Logical Execution Time model on multicores. Martinez et al. [42] introduce also the formulas for the latency computation of sporadic tasks with the implicit communication model. In this work, we also consider the implicit communication, but the tasks activation pattern is periodic. Becker et al. [6, 7, 8] analyze different types of the end-to-end latencies for periodic tasks in abstraction from a concrete scheduling policy as well as for the static off-line schedules [5, 6]. The authors propose an algorithm for an automatic generation of the job-level dependencies guaranteeing that imposed latency constraints are met. The task model in the present paper assumes that the schedule is not static as the tasks' execution times may vary from one task instance to another.

Trigger chains. While this paper studies the data chains, there is also a large body of literature on the trigger chains that are subject to various end-to-end constraints. The holistic analysis [60] applies to the tasks whose execution can be triggered by the arrival of a message or by the completion of a preceding task. Schlatow and Ernst [57] propose an upper bound on the end-to-end latency of task chains with synchronous and asynchronous communicating threads.

Multiframe. The multiframe model [4, 44] or transaction tasks [54, 55, 61] generate cyclically a fixed sequence of different jobs according to a predefined time pattern. They can also be used to express the sequential execution of communicating jobs. However, they do not consider that a job could transfer the data to two or more jobs (linear model [25, 29]) and the delay between the releases of two directly communicating jobs must be constant or upper-bounded by a constant value. In the data chains, this delay can vary from one task instance to another, and the same task can be involved in multiple data flows.

CAN messages. Di Natale et al. [48] propose a method for the evaluation of the worst-case latency for mixed chains of real-time tasks and Controller Area Network (CAN) messages. Zeng et al. [66, 67, 68] describe the use of statistical analysis to compute the probability distribution of end-to-end latencies for CAN message chains. In this work, we consider that a task can be preempted at any point during its execution and we analyze the effects of the preemption on the latency.

7 Conclusion

In this paper, we proposed a polynomial-time upper bound on the worst-case latency of data chains. We showed with benchmark based on the automotive application that the use of the proposed bound can disentangle the complexity of the worst-case latency estimation with little loss in terms of precision. The bound lays the groundwork for solving several problems in software engineering of real-time systems like, for instance, task periods and priorities assignment optimization under latency constraints [9, 16, 17, 35, 39, 69].

Appendix

Proof of Lemma 2. Let $p = (r_1, \ldots, r_n)$ and let $r_1, \ldots, r_n$ be the release time instants of the consecutive instances of tasks that propagate the data such that each instance is released as late as possible and can terminate at its worst-case finishing time. Now suppose that $p' = (r'_1, \ldots, r'_n)$ is an arbitrary path of the same

chain such that $r'_1 = r_1$. First, we prove by induction that $\forall i\, 1 \leq i \leq n : r'_i \leq r_i$. Base case ($i = 1$). By definition $r'_1 \leq r_1$. Induction step ($1 < i \leq n$). By the induction hypothesis: $r'_i \leq r_i$. Since the chain's tasks are schedulable and have their deadlines equal to their periods, every task instance must finish before the release of its next instance. If τ_i is released before r_i at $r'_i < r_i$, then it executes within the interval $[r'_i, r'_i + T_i]$ and must finish before r_i. It fills the input register of τ_{i+1} with the results of its computation before r_i. The instance of task τ_i released at r_i executes within the interval $[r_i, r_i + T_i]$ and fills the register of τ_{i+1} with the results of its computation after r_i. Consequently, no instance of τ_i released at $r'_i \leq r_i$ can write into the register of τ_{i+1} at a later time than τ_i released at r_i does. Hence, if task τ_{i+1} released at r_{i+1} has already data from τ_i released at r_i, then the earlier or the same instance of τ_{i+1} must have the data from τ_i released at r'_i: $r'_{i+1} \leq r_{i+1}$.

From the definition of maximum latency given by Equation (1), the lemma statement holds iff: $f_n(r'_n) \leq f_n(r_n)$. The LHS of the previous expression gives a finishing time of task τ_n released at $r'_n \leq r_n$ and RHS a finishing time of the same task released at r_n. If $r_n = r'_n$, then $f_n(r_n) = f_n(r'_n)$ and the statement is clearly true. Otherwise, if $r'_n < r_n$, then by the same reasoning as in the induction part, every task instance must finish before the release of its next instance, and it follows that $f_n(r'_n) < f_n(r_n)$. □

Acknowledgments

Tomasz Kloda was supported by the Chair for Cyber-Physical Systems in Production Engineering at TUM and the Alexander von Humboldt Foundation.

References

[1] Vallabh Anwikar and Purandar Bhaduri. "Timing analysis of real-time embedded systems using model checking". In: *18th International Conference on Real-Time and Network Systems*. 2010, pp. 119–128.

[2] Neil C Audsley, Alan Burns, Mike F Richardson, and Andy J Wellings. "Real-time scheduling: the deadline-monotonic approach". In: *in Proc. IEEE Workshop on Real-Time Operating Systems and Software*. 1991.

[3] *AUTOSAR Technical Overview*. Release 4.1. Rev. 2, Ver. 1.1.0. The AUTOSAR Consortium. Oct. 2013.

[4] Sanjoy Baruah, Deji Chen, Sergey Gorinsky, and Aloysius Mok. "Generalized Multiframe Tasks". In: *Real-Time Syst.* 17.1 (July 1999), pp. 5–22.

[5] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. "Analyzing End-to-End Delays in Automotive Systems at Various Levels of Timing Information". In: *ACM SIGBED Review: Special Issue on 4th International Workshop on Real-time Computing and Distributed Systems in Emergent Applications* 14.4 (Nov. 2017), pp. 1–6.

[6] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. "End-to-End Timing Analysis of Cause-Effect Chains in Automotive Embedded Systems". In: *Journal of Systems Architecture* 80.Supplement C (Oct. 2017).

[7] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas 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'16, 05 Jul 2016, Toulouse, France*. 2016.

[8] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. "Synthesizing Job-Level Dependencies for Automotive Multi-rate Effect Chains". In: *Proceedings of the 22nd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. Daegu, Korea, Aug. 2016, pp. 159–169.

[9] Matthias Becker and Saad Mubeen. "Timing Analysis Driven Design-Space Exploration of Cause-Effect Chains in Automotive Systems". In: *44th Annual Conference of the IEEE Industrial Electronics Society*. Oct. 2018.

[10] Lucia Lo Bello, Riccardo Mariani, Saad Mubeen, and Sergio Saponara. "Recent Advances and Trends in On-Board Embedded and Networked Automotive Systems". In: *IEEE Trans. Industrial Informatics* 15.2 (2019), pp. 1038–1051.

[11] Enrico Bini and Giorgio C. Buttazzo. "Biasing effects in schedulability measures". In: *16th Euromicro Conference on Real-Time Systems, 2004. ECRTS 2004. Proceedings*. 2014, pp. 196–203.

[12] Enrico Bini, Giorgio C. Buttazzo, and Giuseppe M. Buttazzo. "A Hyperbolic Bound for the Rate Monotonic Algorithm". In: *13th Euromicro Conference on Real-Time Systems (ECRTS 2001), 13-15 June 2001, Delft, The Netherlands, Proceedings*. 2001, pp. 59–66.

[13] Alessandro Biondi and Marco Di Natale. "Achieving Predictable Multicore Execution of Automotive Applications Using the LET Paradigm". In: *Proceedings of the 24th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2018)*. Porto, Portugal, Apr. 2018.

[14] Maxime Chéramy, Pierre-Emmanuel Hladik, and Anne-Marie Déplanche. "SimSo: A Simulation Tool to Evaluate Real-Time Multiprocessor Scheduling Algorithms". In: *Proc. of the 5th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*. WATERS. 2014.

[15] Dayton Clark. "HIC: an operating system for hierarchies of servo loops". In: *Proceedings of the 1989 IEEE International Conference on Robotics and Automation, Scottsdale, Arizona, USA, May 14-19, 1989*. IEEE Computer Society, 1989, pp. 1004–1009.

[16] Abhijit Davare, Qi Zhu, Marco Di Natale, Claudio Pinello, Sri Kanajan, and Alberto L. Sangiovanni-Vincentelli. "Period Optimization for Hard Real-time Distributed Automotive Systems". In: *Proceedings of the 44th Design Automation Conference, DAC 2007, San Diego, CA, USA, June 4-8, 2007*. IEEE, 2007, pp. 278–283.

[17] Peng Deng, Qi Zhu, Abhijit Davare, Anastasios I. Mourikis, Xue Liu, and Marco Di Natale. "An Efficient Control-Driven Period Optimization Algorithm for Distributed Real-Time Systems". In: *IEEE Trans. Computers* 65.12 (2016), pp. 3552–3566.

[18] Guy Durrieu, Madeleine Faugère, Sylvain Girbal, Daniel Gracia Pérez, Claire Pagetti, and W. Puffitsch. "Predictable Flight Management System Implementation on a Multicore Processor". In: *Embedded Real Time Software (ERTS'14)*. Toulouse, France, Feb. 2014.

[19] *EAST-ADL Domain Model Specification*. V2.1.12. EAST-ADL Association. 2013. URL: `http://www.east-adl.info/Specification/V2.1.12/EAST-ADL-Specification_V2.1.12.pdf` (visited on 04/17/2020).

[20] Christof Ebert and John Favaro. "Automotive software". In: *IEEE Software* 3 (2017), pp. 33–39.

[21] Nico Feiertag, Kai Richter, Johan Nordlander, and Jan Jonsson. "A Compositional Framework for End-to-End Path Delay Calculation of Automotive Systems under Different Path Semantics". In: *Proceedings of the IEEE Real-Time System Symposium – Workshop on Compositional Theory and Technology for Real-Time Embedded Systems, Barcelona, Spain, November 30, 2008*. 2008.

[22] Julien Forget, Frédéric Boniol, Emmanuel Grolleau, David Lesens, and Claire Pagetti. "Scheduling Dependent Periodic Tasks without Synchronization Mechanisms". In: *Proceedings of the 16th IEEE Real-Time and Embedded Technology and Applications Symposium*. Stockholm, Sweden, Apr. 2010, pp. 301–310.

[23] Julien Forget, Frédéric Boniol, David Lesens, and Claire Pagetti. "A Real-time Architecture Design Language for Multi-rate Embedded Control Systems". In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. SAC '10. Sierre, Switzerland: ACM, 2010, pp. 527–534.

[24] Julien Forget, Frédéric Boniol, and Claire Pagetti. "Verifying end-to-end real-time constraints on multi-periodic models". In: *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Sept. 2017, pp. 1–8.

[25] J. J. Gutiérrez García, José C. Palencia Gutiérrez, and Michael González Harbour. "Schedulability analysis of distributed hard real-time systems with multiple-event synchronization". In: *12th Euromicro Conference on Real-Time Systems (ECRTS 2000), 19-21 June 2000, Stockholm, Sweden, Proceedings*. IEEE Computer Society, 2000, pp. 15–24.

[26] Richard Gerber, Seongsoo Hong, and Manas Saksena. "Guaranteeing end-to-end timing constraints by calibrating intermediate processes". In: *1994 Proceedings Real-Time Systems Symposium*. Dec. 1994, pp. 192–203.

[27] Richard Gerber, Seongsoo Hong, and Manas Saksena. "Guaranteeing Real-Time Requirements With Resource-Based Calibration of Periodic Processes". In: *IEEE Transactions on Software Engineering* 21.7 (July 1995), pp. 579–592.

[28] Joël Goossens and Christophe Macq. "Limitation of the Hyper-Period in Real-Time Periodic Task Set Generation". In: *In Proceedings of the RTS Embedded System (RTS'01)*. 2001, pp. 133–147.

[29] José C. Palencia Gutiérrez, José Javier Gutiérrez García, and Michael González Harbour. "On the schedulability analysis for distributed hard real-time systems". In: *Proceedings of the Ninth Euromicro Workshop on Real-Time Systems, RTS 1997, 11-13 June, 1997, Toledo, Spain*. IEEE Computer Society, 1997, pp. 136–143.

[30] Arne Hamann, Dakshina Dasari, Simon Kramer, Michael Pressler, and Falk Wurst. "Communication Centric Design in Complex Automotive Embedded Systems". In: *29th Euromicro Conference on Real-Time Systems (ECRTS)*. Vol. 76. Leibniz International Proceedings in Informatics (LIPIcs). Dubrovnik, Croatia, 2017, 10:1–10:20.

[31] C-C Han and H-Y Tyan. "A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms". In: *Proceedings Real-Time Systems Symposium*. Dec. 1997, pp. 36–45.

[32] Ross Honsberger. *Mathematical Gems II*. Mathematical Association of America, 1976, pp. 54–57.

[33] Mathai Joseph and Paritosh K. Pandya. "Finding Response Times in a Real-Time System". In: *Comput. J.* 29.5 (1986), pp. 390–395.

[34] Jad Khatib, Alix Munier-Kordon, Enagnon Cedric Klikpo, and Kods Trabelsi-Colibet. "Computing Latency of a Real-time System Modeled by Synchronous Dataflow Graph". In: *Proceedings of the 24th International Conference on Real-Time Networks and Systems*. RTNS '16. Brest, France: ACM, 2016, pp. 87–96.

[35] Tobias Klaus, Florian Franzmann, Matthias Becker, and Peter 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*. RTNS '18. Chasseneuil-du-Poitou, France: ACM, 2018, pp. 93–103.

[36] Tomasz Kloda, Antoine Bertout, and Yves Sorel. "Latency analysis for data chains of real-time periodic tasks". In: *23rd IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2018, Torino, Italy, September 4-7, 2018*. Sept. 2018.

[37] Donald E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997.

[38] Simon Kramer, Dirk Ziegenbein, and Arne Hamann. "Real World Automotive Benchmark For Free". In: *6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*. July 2015.

[39] Lukas Krawczyk, Mahmoud Bazzal, Ram Prasath Govindarajan, and Carsten Wolff. "Model-Based Timing Analysis and Deployment Optimization for Heterogeneous Multi-core Systems using Eclipse APP4MC". In: *22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS Companion 2019, Munich, Germany, September 15-20, 2019*. Ed. by Loli Burgueño et al. IEEE, 2019, pp. 44–53.

[40] Joseph Y.-T. Leung and M. L. Merrill. "A Note on Preemptive Scheduling of Periodic, Real-Time Tasks". In: *Inf. Process. Lett.* 11.3 (1980), pp. 115–118.

[41] C. L. Liu and James W. Layland. "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment". In: *J. ACM* 20.1 (1973), pp. 46–61.

[42] Jorge Martinez, Ignacio Sañudo, Paoloa Burgio, and Marko Bertogna. "End-to-end latency characterization of implicit and LET communication models". In: *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS), Dubrovnik, Croatia*. 2017.

[43] Morteza Mohaqeqi, Mitra Nasri, Yang Xu, Anton Cervin, and Karl-Erik AArzén. "On the Problem of Finding Optimal Harmonic Periods". In: *Proceedings of the 24th International Conference on Real-Time Networks and Systems*. RTNS '16. Brest, France: ACM, 2016, pp. 171–180.

[44] Aloysius K. Mok and Deji Chen. "A multiframe model for real-time tasks". In: *Proceedings of the 17th IEEE Real-Time Systems Symposium (RTSS '96), December 4-6, 1996, Washington, DC, USA*. IEEE Computer Society, 1996, pp. 22–29.

[45] Saad Mubeen, Jukka Mäki-Turja, and Mikael Sjödin. "Support for End-to-End Response-Time and Delay Analysis in the Industrial Tool Suite: Issues, Experiences and a Case Study". In: *Computer Science and Information Systems, ISSN: 1820-0214* 10.1 (Jan. 2013), pp. 453–482.

[46] Saad Mubeen, Thomas Nolte, Mikael Sjödin, John Lundbäck, and Kurt-Lennart Lundbäck. "Supporting timing analysis of vehicular embedded systems through the refinement of timing constraints". In: *Software and Systems Modeling* 18.1 (2019), pp. 39–69.

[47] Marco Di Natale, Paolo Giusto, Sri Kanajan, Claudio Pinello, and Patrick Popp. "Architecture Exploration for Time-Critical and Cost-Sensitive Distributed Systems". In: *SAE Technical Paper*. SAE International, Apr. 2007.

[48] Marco Di Natale, Haibo Zeng, Paolo Giusto, and Arkadeb Ghosal. *Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice*. Springer Publishing Company, Incorporated, 2012.

[49] Claire Pagetti. "Programming mutli/many-core COTS for critical embedded systems". Habilitation à diriger des recherches. INPT, Jan. 2015.

[50] Claire Pagetti, Julien Forget, Frédéric Boniol, Mikel Cordovilla, and David Lesens. "Multi-task Implementation of Multi-periodic Synchronous Programs". In: *Discrete Event Dynamic Systems* 21.3 (2011), pp. 307–338.

[51] Paolo. Pazzaglia, Alessandro Biondi, and Marco Di Natale. "Simple and General Methods for Fixed-Priority Schedulability in Optimization Problems". In: *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2019, pp. 1543–1548.

[52] Rodolfo Pellizzoni and Giuseppe Lipari. "A New Sufficient Feasibility Test for Asynchronous Real-Time Periodic Task Sets". In: *16th Euromicro Conference on Real-Time Systems (ECRTS), Catania, Italy*. 2004, pp. 204–211.

[53] Rodolfo Pellizzoni and Giuseppe Lipari. “Feasibility Analysis of Real-Time Periodic Tasks with Offsets”. In: *Real-Time Systems* 30.1-2 (2005), pp. 105–128.

[54] Rodolfo Pellizzoni and Giuseppe Lipari. “Holistic analysis of asynchronous real-time transactions with earliest deadline scheduling”. In: *Journal of Computer and System Sciences* 73.2 (2007). Special Issue: Real-time and Embedded Systems, pp. 186–206.

[55] Ahmed Rahni, Emmanuel Grolleau, Michaël Richard, and Pascal Richard. “Feasibility Analysis of Real-time Transactions”. In: *Real-Time Syst.* 48.3 (May 2012), pp. 320–358.

[56] A. C. Rajeev, Swarup Mohalik, Manoj G. Dixit, Devesh 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*. EMSOFT ’10. Scottsdale, Arizona, USA: ACM, 2010, pp. 129–138.

[57] Johannes Schlatow and Rolf Ernst. “Response-Time Analysis for Task Chains in Communicating Threads”. In: *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. Apr. 2016, pp. 1–10.

[58] Lui Sha, Ragunathan Rajkumar, and John P Lehoczky. “Priority Inheritance Protocols: An Approach to Real-Time Synchronization”. In: *IEEE Transactions on computers* 39.9 (1990), pp. 1175–1185.

[59] *Timing Augmented Description Language (TADL2) syntax, semantics, metamodel*. Ver. 2, Deliverable 11. Rev. 2, Ver. 1.1.0. Aug. 2012.

[60] Ken Tindell and John Clark. “Holistic Schedulability Analysis for Distributed Hard Real-time Systems”. In: *Microprocess. Microprogram.* 40.2-3 (Apr. 1994), pp. 117–134.

[61] Karim Traore, Emmanuel Grolleau, and Francis Cottet. “Characterization and Analysis of Tasks with Offsets: Monotonic Transactions”. In: *12th IEEE Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2006), 16-18 August 2006, Sydney, Australia*. IEEE Computer Society, 2006, pp. 10–16.

[62] Randy Walter. “Digital Avionics Handbook, Second Edition - 2 Volume Set”. In: Electrical Engineering Handbook. CRC Press, Dec. 2000. Chap. Flight Management Systems.

[63] Rémy Wyss, Frédéric Boniol, Julien Forget, and Claire Pagetti. “Propriétés de latence, fraîcheur et réactivité dans un programme synchrone multi-périodique”. In: *Approches Formelles dans l’Assistance au Développement de Logiciels*. Nancy, France, Apr. 2013.

[64] Rémy Wyss, Frédéric Boniol, Claire Pagetti, and Julien Forget. “End-to-end Latency Computation in a Multi-periodic Design”. In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. SAC ’13. Coimbra, Portugal: ACM, 2013, pp. 1682–1687.

[65] Yang Xu, Anton Cervin, and Karl-Erik Årzén. “Harmonic Scheduling and Control Co-design”. In: *2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. Aug. 2016, pp. 182–187.

[66] Haibo Zeng. “Probabilistic Timing Analysis of Distributed Real-time Automotive Systems”. PhD thesis. EECS Department, University of California, Berkeley, Dec. 2008.

[67] Haibo Zeng, Marco Di Natale, Paolo Giusto, and Alberto Sangiovanni-Vincentelli. “Stochastic Analysis of CAN-Based Real-Time Automotive Systems”. In: *IEEE Transactions on Industrial Informatics* 5.4 (Nov. 2009), pp. 388–401.

[68] Haibo Zeng, Marco Di Natale, Paolo Giusto, and Alberto L. Sangiovanni-Vincentelli. “Statistical analysis of Controller Area Network message response times”. In: *IEEE Fourth International Symposium on Industrial Embedded Systems, SIES 2009, Ecole Polytechnique Federale de Lausanne, Switzerland, July 8-10, 2009*. 2009, pp. 1–10.

[69] Yecheng Zhao, Vinit Gala, and Haibo Zeng. “A Unified Framework for Period and Priority Optimization in Distributed Hard Real-Time Systems”. In: *IEEE Trans. on CAD of Integrated Circuits and Systems* 37.11 (2018), pp. 2188–2199.