Efficient Latency Bound Analysis for Data Chains of Real-Time Tasks in Multiprocessor Systems

Jiankang Ren*,§, Xin He*, Junlong Zhou†, Hongwei Ge*, Guowei Wu‡, and Guozhen Tan*
*School of Computer Science and Technology, Dalian University of Technology, China
†School of Computer Science and Engineering, Nanjing University of Science and Technology, China
‡School of Software Technology, Dalian University of Technology, China

***Abstract*—End-to-end latency analysis is one of the key problems in the automotive embedded system design. In this paper, we propose an efficient worst-case end-to-end latency analysis method for data chains of periodic real-time tasks executed on multiprocessors under a partitioned fixed-priority preemptive scheduling policy. The key idea of this research is to improve the analysis efficiency by transforming the problem of bounding the worst-case latency of the data chain to a problem of bounding the releasing interval of data propagation instances for each pair of consecutive tasks in the chain. In particular, we derive an upper bound on the releasing interval of successive data propagation instances to yield the desired data chain latency bound by a simple accumulation. Based on the above idea, we present an efficient latency upper bound analysis algorithm with polynomial time complexity. Experiments with randomly generated task sets based on a generic automotive benchmark show that our proposed approach can obtain a relatively tighter data chain latency upper bound with lower computational cost.**

I. Introduction

With the utilization of sophisticated hybrid engines, applications of driving assistance and autonomous driving systems, automotive systems are increasingly incorporating more functional components that are tightly coupled via intensive data sharing. Given this complexity, the automotive systems usually involves some task chains from the sensor to the actuator in the computational processing to fulfill certain functionalities through end-to-end computations [1]–[4]. For most of safety-relevant automotive applications, there are typically some strict timing constraints on the data propagation through the task chain, thereby complying with the functional safety automotive standard ISO 26262 [5]. Moreover, the rapid development of chip technology leads to an increasing trend of deploying automotive systems on highly-integrated multiprocessor platforms to achieve high computing power and simplify the integration of functionality [6]. Consequently, it is crucial to provide an efficient end-to-end timing analysis strategy for task chains in multiprocessor systems, such that the strict timing requirements on end-to-end computations can be verified efficiently for the complex automotive systems.

Since Tindell et al. [7] presented the pioneer work on the end-to-end time analysis for event-triggered paths of tasks, a rich literature on the end-to-end timing analysis of real-time systems has developed. Traditionally, work in this domain has focused on uniprocessor platforms [8], but recently efforts have shifted towards multiprocessor platforms. The end-to-end timing analysis of task chains in multiprocessor platforms can be generally categorized in two classes: *holistic response-time analysis* [9], [10] and *end-to-end latency analysis* [11], [12]. Holistic response time analysis calculates the upper bound on the response time of the last task in the task chain to verify the end-to-end deadline requirements. In contrast, the end-to-end latency analysis derives the maximum time taken by the data propagation through the task chain from a sensor to an actuator to check the control performance requirements. In this paper, we focus on the worst-case end-to-end latency analysis for the data chains of periodic real-time tasks in multiprocessor systems. Note that, it is not trivial to efficiently calculate the exact upper bound of the end-to-end latency for such systems, since the problem of over- and under-sampling will appear when the tasks in a task chain are activated with different periodicities. For the data chains of periodic tasks running on different nodes, Davare et al. [11] proposed a worst-case latency analysis approach with linear-time complexity, which calculates the worst-case end-to-end latency of a task chain by simply adding the worst-case response times and the periods of all the tasks in the chain. However, this method is over-pessimistic, since it separately considers the timing behavior of each task in a task chain, while neglecting the temporal decoupling of execution and communication in the chain. To this end, Kloda et al. [12] proposed a worst-case latency analysis method based on closed-form expression for communicating real-time tasks, and it can obtain much more accurate estimates of the worst-case latency. However, this method is not scalable due to the exponential time complexity, and its high complexity and execution time may become huge drawbacks in interactive real-time system design environments in which the latency analysis algorithm should be called a large number of times during a process of interactive system design and rapid system prototyping. To solve this problem, we propose a polynomial time algorithm for computing efficiently relatively tighter, rather than exact, upper bounds for the end-to-end latency. Our approach, called δ-Bound, is based on the analysis of upper bounds on the releasing interval of successive data propagation instances in the data chain. On the basis of the releasing interval upper bounds of data propagation instances for each pair of consecutive tasks in the chain, we can efficiently compute the latency upper bound by a simple accumulation without enumerating all the release time instants

§The corresponding author: Dr. Jiankang Ren, rjk@dlut.edu.cn.

978-3-9819263-4-7/DATE20/©2020 EDAA

of the first task in the chain within the hyperperiod as in [12].

Contributions. This paper makes the following contributions:

- We reduced the problem of bounding the worst-case latency of the data chain to a problem of bounding the releasing interval of data propagation instances for each pair of consecutive tasks in the data chain.
- We derived an upper bound on the releasing interval of successive data propagation instances to yield the desired data chain latency bound.
- We developed an efficient algorithm to compute the worst-case latency bound of the data chain with polynomial time complexity if the worst-case response times of tasks are known.

Our evaluation shows that our proposed analysis approach has very high efficiency with low precision loss.

II. System Model

The real-time system considered in this paper consists of n independent periodic real-time tasks, identified by $\Gamma = \{\tau_1,...,\tau_n\}$, and scheduled on a multiprocessor platform with m identical, unit-capacity processors, denoted as $\Pi = \{\pi_1,...,\pi_m\}$, according to a partitioned fixed-priority preemptive scheduling policy. Each task $\tau_i \in \Gamma$ is characterized by a triple $\tau_i = (C_i, T_i, D_i)$, where

- C_i is the worst case execution time (WCET) of task τ_i;
- T_i is the period of task τ_i; and
- D_i is the relative deadline of task τ_i.

For a task τ_i, we use the term *utilization* denoted by U_i to represent the ratio C_i/T_i, and let U_Γ denote the *total utilization* of task set Γ: $\mathsf{U}_\Gamma = \sum_{\tau_i \in \Gamma} U_i$. We assume that all tasks are implicit deadline tasks (i.e., $D_i = T_i$), and they are initially released simultaneously at time instant $t = 0$. The instance release time of task τ_i is denoted as r_i, and it belongs to an infinite set $A(\tau_i) = \{0, T_i, 2T_i, ...\}$. For an instance of task τ_i released at r_i, its finishing time is defined as $f_i(r_i)$. The worst-case response time (WCRT) of a task τ_i is denoted as $\mathcal{R}_i$, which is the longest time interval between its instance release time and the finishing time of its execution with considering interference from other tasks in the system. For each task $\tau_i \in \Gamma$, if its WCRT is not larger than its deadline (i.e., $\mathcal{R}_i \leq D_i$), the system is considered schedulable. Each task $\tau_i \in \Gamma$ is assumed to have a unique priority P_i. If τ_i has a higher priority than τ_j, we denote it as $P_i > P_j$. Moreover, we suppose that the task-to-processor assignment is already given.

In this paper, we consider the systems where the tasks are executed on the read-execute-write model. In this model, a task firstly copies the data from the input register to its local environment before starting its execution, and then uses only the local copied variants during its execution. After the execution, the produced results are written to another register which might be read by another task. In AUTOSAR, this model is defined as the implicit communication [2], [13], and it can be modeled as a *data chain* F_n ($n \geq 1$) [11], [12]. A data chain F_n is a sequence of $n \geq 1$ tasks $(\tau_1, \tau_2, ...\tau_n)$ describing a flow of communication between the tasks realized via shared registers. The first and the last tasks of a data chain F_n are

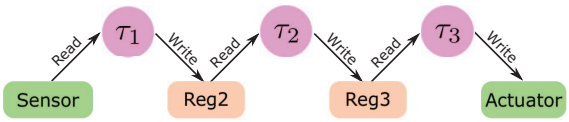


Fig. 1: An example data chain with three tasks.

denoted by $head(F_n) = \tau_1$ and $last(F_n) = \tau_n$, respectively. If $n = 1$, the task τ_1 calculates its outputs and writes them to the appropriate actuator after it reads the data from its sensor. If $n \geq 2$, the data chain F_n processes the data as follows:

- The first task τ_1 reads the data from a sensor, computes the results, and writes these results into the register shared with task τ_2 at the end of its execution;
- Task τ_i ($1< i < n$) reads the data from the register shared with task τ_{i-1}, produces results and writes them into the register shared with task τ_{i+1} at the end of its execution;
- The last task τ_n reads the data from the register shared with task τ_{n-1}, computes the outputs and writes them to the corresponding actuator at the end of its execution.

Note that in such a communication model, there is no signaling between tasks, and tasks are scheduled according to the assigned priorities only. As shown in Fig. 1, it is an example data chain with three tasks τ_1, τ_2 and τ_3 executed on the read-execute-write model.

To model the end-to-end latency of a data chain F_n, we use the same notions as in [12], which is based on the concept of the timed path introduced in [8].

Definition II.1. (**Data Propagation Paths** [12]). *Let F_n be a data chain whose first task $\tau_1 = head(F_n)$ is released at time instant $r_1 \in A(\tau_1)$, a set of* data propagation paths*, denoted by $\Omega(F_n, r_1)$ is defined as a set of $\varphi = (r_1, ..., r_n)$, where $r_i \in \varphi$ is the release time of the task τ_i instance that propagates the data, and task τ_i propagates a data when it writes this data for the first time into a register shared with the next task τ_{i+1}.*

Definition II.2. (**Data Chain Latency** [12]). *Let F_n be a data chain whose registers are initially empty. Its* data chain latency*, denoted as $L(F_n)$, 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.*

Definition II.3. (**Maximum Data Chain Latency at** r_1 [12]). *Let F_n be a data chain of $n \geq 1$ tasks, its first task $\tau_1 = head(F_n)$ is released at $r_1 \in A(\tau_1)$, and the input of task τ_1 arrives in its register at r_1. The maximum latency of data chain F_n released at r_1 is not greater than:*

$$L(F_n, r_1) = \max_{\varphi \in \Omega(F_n, r_1)} f_n(r_n) - r_1. \tag{1}$$

where $r_n \in \varphi$, $\tau_n = last(F_n)$ and $f_n(r_n)$ is the finishing time of task τ_n released at r_n.

Consider a data chain $F_3 = (\tau_1, \tau_2, \tau_3)$, where $\tau_1 = (C_1 = 1, T_1 = 5, D_1 = 5)$, $\tau_2 = (C_2 = 1, T_2 = 8, D_2 = 8)$, $\tau_3 = (C_3 = 3, T_3 = 5, D_3 = 5)$, and $P_1 > P_2 > P_3$. As

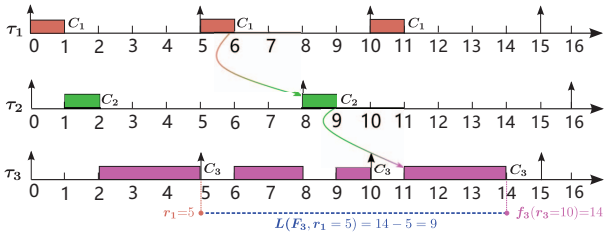


Fig. 2: An execution scenario of a data chain with three tasks.

shown in Fig. 2, it illustrates the data chain latency for a data propagation path by giving an execution scenario of F_3.

For a data chain F_n, if the data arrives at the sensor immediately after an instance release of its first task $\tau_1 = head(F_n)$, the data may need to be handled by the next task instance. In the worst case, the delay experienced by the data in the sensor register is at most one period of the first task τ_1. Therefore, by considering the sensor data detection delay of the first task in the chain and all possible release time instants of the first task, we can obtain the *worst-case end-to-end latency* of the data chain F_n as follows [12]:

$$L(F_n) < T_1 + \max_{r_1 \in A(\tau_1)} L(F_n, r_1). \tag{2}$$

Problem: The main objective of this work is to design an efficient worst-case end-to-end latency analysis strategy for data chains of periodic real-time tasks with implicit deadlines executed by a partitioned fixed-priority preemptive scheduler upon a multiprocessor platform, such that a tight upper bound for the data chain latency can be obtained with lower computational cost. In this paper, we reduce the problem of bounding the worst-case latency of data chains to a problem of bounding the releasing interval of data propagation instances for each pair of consecutive tasks in the chain. Based on such problem reduction, we also present a polynomial time algorithm to calculate the latency upper bounds with low precision loss.

III. Worst-Case Data Chain Latency Analysis

In order to improve the analysis efficiency of the worst-case end-to-end latency for data chains, we reduce the problem of bounding the worst-case latency of data chains to a problem of bounding the releasing interval of successive data propagation instances (see Theorem III.1). Moreover, we derive an upper bound on the releasing interval of successive data propagation instances to yield the desired data chain latency bound (see Theorem III.2). Before introducing our proposed analysis algorithm, we first give some related definitions.

Definition III.1. *Let $F_n = (\tau_1, \tau_2, ..., \tau_n)$ be a data chain of $n \geq 2$ tasks and $\varphi = (r_1, r_2, ..., r_n)$ be a data propagation path of F_n, the interval between the propagation instance release times of two adjacent tasks τ_i and τ_{i+1} $(1 \leq i \leq n-1)$ in the path φ, denoted by $\delta_{i,i+1}$, is defined as,*

$$\delta_{i,i+1} = r_{i+1} - r_i. \tag{3}$$

where $r_i \in \varphi$ and $r_{i+1} \in \varphi$.

Definition III.2. *For a data chain $F_n = (\tau_1, \tau_2, ..., \tau_n)$ of $n \geq 2$ tasks, let $\delta^{\mathsf{ub}}_{i,i+1}$ denote the upper bound of the interval between the propagation instance release times of two consecutive tasks τ_i and τ_{i+1} $(1 \leq i \leq n-1)$ for any data propagation path of the chain F_n.*

It is straightforward to observe that

$$\delta^{\mathsf{ub}}_{i,i+1} \geq \max_{\substack{r_1 \in A(\tau_1), \varphi \in \Omega(F_n, r_1), \\ r_i \in \varphi, r_{i+1} \in \varphi}} (r_{i+1} - r_i). \tag{4}$$

Based upon the above definitions, we obtain the following alternative representation of the worst-case end-to-end latency bound for the data chain.

Theorem III.1. *Let F_n be a data chain of $n \geq 2$ tasks, its worst-case end-to-end latency $L(F_n)$ is bounded by:*

$$L^{\mathsf{ub}}(F_n) = T_1 + \mathcal{R}_n + \sum_{i=1}^{n-1} \delta^{\mathsf{ub}}_{i,i+1}, \tag{5}$$

where T_1 is the period of task $\tau_1 = head(F_n)$, $\mathcal{R}_n$ is the WCRT of task $\tau_n = last(F_n)$, and $\delta^{\mathsf{ub}}_{i,i+1}$ is the upper bound of the interval between the propagation instance release times of consecutive tasks τ_i and τ_{i+1} in the chain F_n.

Proof. By equations (1), (2) and (4), we have

$$\begin{aligned}
L(F_n) &< T_1 + \max_{r_1 \in A(\tau_1)} L(F_n, r_1) \\
&= T_1 + \max_{r_1 \in A(\tau_1)} \left(\max_{\varphi \in \Omega(F_n, r_1)} (f_n(r_n) - r_1) \right) \\
&\leq T_1 + \max_{r_1 \in A(\tau_1), \varphi \in \Omega(F_n, r_1)} (r_n + \mathcal{R}_n - r_1) \\
&= T_1 + \mathcal{R}_n + \max_{r_1 \in A(\tau_1), \varphi \in \Omega(F_n, r_1)} \sum_{i=1}^{n-1} \delta_{i,i+1} \\
&\leq T_1 + \mathcal{R}_n + \sum_{i=1}^{n-1} \max_{r_1 \in A(\tau_1), \varphi \in \Omega(F_n, r_1)} \delta_{i,i+1} \\
&\leq T_1 + \mathcal{R}_n + \sum_{i=1}^{n-1} \delta^{\mathsf{ub}}_{i,i+1} \\
&= L^{\mathsf{ub}}(F_n).
\end{aligned}$$

Therefore, $L(F_n)$ is bounded by equation (5). □

Based on Theorem III.1, we reduce the problem of bounding the worst-case latency of a data chain to a problem of bounding the releasing interval of data propagation instances for each pair of consecutive tasks in the chain. In order to derive a bound on the releasing interval of successive data propagation instances, we model each pair of directly communicating tasks τ_i and τ_{i+1} $(1 \leq i \leq n-1)$ in the data chain F_n as producer and consumer. Now, we first present the following lemma to describe the releasing interval of adjacent instances for any two periodic real-time tasks.

Lemma III.1. *For any two tasks $\tau_i = (C_i, T_i, D_i)$ and $\tau_j = (C_j, T_j, D_j)$ that are initially released simultaneously at the time instant 0, let r_i be an instance release time of task τ_i, r_j^{front} be the latest instance release time of task τ_j released no later than r_i, and $\lambda_{i,j} = r_i - r_j^{\mathsf{front}}$, and we can get:*

$$\lambda_{i,j} \in \Lambda_{i,j} = \{0, \eta_{i,j}, ..., T_j - \eta_{i,j}\} \tag{6}$$

where $\eta_{i,j} = \mathsf{gcd}(T_i, T_j)$ that is the greatest common divisor of T_i and T_j.

Proof. We let $T_i = M \cdot \eta_{i,j}$ and $T_j = N \cdot \eta_{i,j}$, where $M \in \mathbb{Z}_{\geq 1}$ and $N \in \mathbb{Z}_{\geq 1}$. Because $\eta_{i,j}$ is the greatest common divisor

of T_i and T_j, we can get $\gcd(M,N) = 1$. Since τ_i and τ_j are initially released simultaneously at the time instant 0, $\lambda_{i,j} = r_i - r_j^{\mathsf{front}} \in \{\mathsf{mod}(x \cdot T_i, T_j) | x \in \mathbb{Z}_{\geq 0} \text{ and } x \leq N\}$. Moreover, $\gcd(M,N) = 1$, and thus $\lambda_{i,j} \in \Lambda_{i,j} = \{\mathsf{mod}(x \cdot T_i, T_j) | x \in \mathbb{Z}_{\geq 0} \text{ and } x \leq N\} = \{\mathsf{mod}(x \cdot M \cdot \eta_{i,j}, N \cdot \eta_{i,j}) | x \in \mathbb{Z}_{\geq 0} \text{ and } x \leq N\} = \{0, \eta_{i,j}, ..., T_j - \eta_{i,j}\}$. □

First, we consider that the consumer τ_{i+1} has a higher priority than the producer τ_i (i.e., $P_{i+1} > P_i$), and τ_i and τ_{i+1} are assigned to the same processor (i.e., $\pi(\tau_i) = \pi(\tau_{i+1})$). Fig. 3 illustrates a sample execution of such two tasks. Since the consumer τ_{i+1} has a higher priority, it can preempt the producer τ_i once it is released. However, the consumer τ_{i+1} cannot read the fresh data from the register shared with the producer τ_i before the completion of the producer τ_i. Therefore, at worst, the fresh data is read for the first time by the earliest instance of consumer τ_{i+1} released after the finishing time $f_i(r_i)$ of producer τ_i. Thus, we can obtain:

$$\delta_{i,i+1} = r_{i+1} - r_i \leq \left\lceil \frac{f_i(r_i)}{T_{i+1}} \right\rceil T_{i+1} - r_i. \tag{7}$$

Lemma III.2. *In a schedulable task set with two communication tasks τ_i and τ_{i+1} assigned to the same processor such that the consumer task τ_{i+1} has a higher priority than the producer task τ_i, the interval $\delta_{i,i+1}$ between the instance release time $r_i \in A(\tau_i)$ of τ_i and the release time r_{i+1} of the instance of τ_{i+1} reading for the first time the data propagated by the instance of τ_i released at r_i is not larger than:*

$$\delta_{i,i+1}^{\mathsf{ub}} = \begin{cases} \mathcal{R}_i + T_{i+1} - \eta_{i,i+1} & \mathsf{mod}(\mathcal{R}_i, \eta_{i,i+1}) = 0 \\ \mathcal{R}_i + T_{i+1} - \mathsf{mod}(\mathcal{R}_i, \eta_{i,i+1}) & \text{Otherwise} \end{cases} \tag{8}$$

Proof. By equation (7) and Lemma III.1, we have

$$\begin{aligned}
&\max_{\substack{r_1 \in A(\tau_1) \\ \varphi \in \Omega(F_n, r_1)}} \delta_{i,i+1} \leq \max_{\substack{r_1 \in A(\tau_1) \\ \varphi \in \Omega(F_n, r_1)}} \left(\left\lceil \frac{f_i(r_i)}{T_{i+1}} \right\rceil T_{i+1} - r_i \right) \\
&\leq \max_{\substack{r_1 \in A(\tau_1) \\ \varphi \in \Omega(F_n, r_1)}} \left(\left\lceil \frac{r_i + \mathcal{R}_i}{T_{i+1}} \right\rceil T_{i+1} - r_i \right) \\
&= \max_{\substack{r_1 \in A(\tau_1) \\ \varphi \in \Omega(F_n, r_1)}} \left(\left\lceil \frac{r_{i+1}^{\mathsf{front}} + \lambda_{i,i+1} + \mathcal{R}_i}{T_{i+1}} \right\rceil T_{i+1} - \left(r_{i+1}^{\mathsf{front}} + \lambda_{i,i+1} \right) \right) \\
&\leq \max_{\substack{\lambda_{i,i+1} \\ \in \Lambda_{i,i+1}}} \left(\left\lceil \frac{\lambda_{i,i+1} + \mathcal{R}_i}{T_{i+1}} \right\rceil T_{i+1} - \lambda_{i,i+1} \right) \\
&= \max_{\substack{\lambda_{i,i+1} \\ \in \Lambda_{i,i+1}}} \left(\left\lceil \frac{\lambda_{i,i+1} + \mathcal{R}_i - \mathsf{mod}(\mathcal{R}_i, T_{i+1}) + \mathsf{mod}(\mathcal{R}_i, T_{i+1})}{T_{i+1}} \right\rceil T_{i+1} \right. \\
&\qquad \left. - \lambda_{i,i+1} \right) \\
&= \max_{\substack{\lambda_{i,i+1} \\ \in \Lambda_{i,i+1}}} \left(\mathcal{R}_i + \left\lceil \frac{\lambda_{i,i+1} + \mathsf{mod}(\mathcal{R}_i, T_{i+1})}{T_{i+1}} \right\rceil T_{i+1} \right. \\
&\qquad \left. - \mathsf{mod}(\mathcal{R}_i, T_{i+1}) - \lambda_{i,i+1} \right) \\
&= \begin{cases} \mathcal{R}_i + T_{i+1} - \eta_{i,i+1} & \mathsf{mod}(\mathcal{R}_i, \eta_{i,i+1}) = 0 \\ \mathcal{R}_i + T_{i+1} - \mathsf{mod}(\mathcal{R}_i, \eta_{i,i+1}) & \text{Otherwise} \end{cases}
\end{aligned}$$

Therefore, $\delta_{i,i+1}$ is bounded by equation (8) when the consumer τ_{i+1} has a higher priority than the producer τ_i. □

Now, we consider the producer τ_i has a higher priority than the consumer τ_{i+1} (i.e., $P_i > P_{i+1}$) and they are assigned to the same processor (i.e., $\pi(\tau_i) = \pi(\tau_{i+1})$). Fig. 4 illustrates a sample execution of such two tasks. Since the instance of the consumer τ_{i+1} that starts before the instance release time r_i of the producer τ_i may not be able to read the data written by τ_i, in the worst case, the data is read for the first time by the first consumer instance released at or after the release time r_i of producer τ_i. Thus, we can obtain:

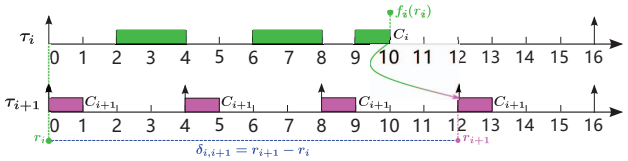


Fig. 3: Two consecutive tasks: τ_{i+1} has a higher priority.

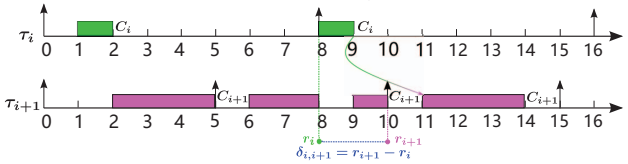


Fig. 4: Two consecutive tasks: τ_i has a higher priority.

$$\delta_{i,i+1} = r_{i+1} - r_i \leq \left\lceil \frac{r_i}{T_{i+1}} \right\rceil T_{i+1} - r_i. \tag{9}$$

Lemma III.3. *In a schedulable task set with two communication tasks τ_i and τ_{i+1} assigned to the same processor such that τ_i writes the results of its computation to the register shared with τ_{i+1} and τ_i has a higher priority than τ_{i+1}, the interval $\delta_{i,i+1}$ between the instance release time $r_i \in A(\tau_i)$ of τ_i and the release time r_{i+1} of the instance of τ_{i+1} reading for the first time the data propagated by the instance of τ_i released at r_i is not larger than:*

$$\delta_{i,i+1}^{\mathsf{ub}} = T_{i+1} - \eta_{i,i+1}. \tag{10}$$

Proof. By equation (9) and Lemma III.1, we have

$$\begin{aligned}
&\max_{\substack{r_1 \in A(\tau_1) \\ \varphi \in \Omega(F_n, r_1)}} \delta_{i,i+1} = \max_{\substack{r_1 \in A(\tau_1) \\ \varphi \in \Omega(F_n, r_1)}} \left(\left\lceil \frac{r_i}{T_{i+1}} \right\rceil T_{i+1} - r_i \right) \\
&= \max_{\substack{r_1 \in A(\tau_1) \\ \varphi \in \Omega(F_n, r_1)}} \left(\left\lceil \frac{r_{i+1}^{\mathsf{front}} + \lambda_{i,i+1}}{T_{i+1}} \right\rceil T_{i+1} - \left(r_{i+1}^{\mathsf{front}} + \lambda_{i,i+1} \right) \right) \\
&\leq \max_{\lambda_{i,i+1} \in \Lambda_{i,i+1}} \left(\left\lceil \frac{\lambda_{i,i+1}}{T_{i+1}} \right\rceil T_{i+1} - \lambda_{i,i+1} \right) \\
&\leq T_{i+1} - \eta_{i,i+1}.
\end{aligned}$$

Therefore, $\delta_{i,i+1}$ is bounded by equation (10) when the producer τ_i has a higher priority than the consumer τ_{i+1}. □

Following the similar reasoning to the one presented for the case that the consumer τ_{i+1} has a higher priority than the producer τ_i, we can obtain the following lemma for a chain of tasks allocated to different processors.

Lemma III.4. *In a schedulable task set with two communication tasks τ_i and τ_{i+1} assigned to different processors, the interval $\delta_{i,i+1}$ between the instance release time $r_i \in A(\tau_i)$ of τ_i and the release time r_{i+1} of the instance of τ_{i+1} reading for the first time the data propagated by the instance of τ_i released at r_i is bounded by equation (8).*

Proof. Since tasks τ_i and τ_{i+1} are assigned to different processors, they cannot block each other. Thus, in the worst case,

the data is read for the first time by the earliest instance of task τ_{i+1} released after the completion of task τ_i (i.e., equation (7)). Therefore, the releasing interval of data propagation instances for τ_i and τ_{i+1} is bounded by equation (8). □

From Lemmas III.2, III.3, and III.4, we can obtain the following theorem.

Theorem III.2. *In a schedulable task set with two communication tasks τ_i and τ_{i+1} such that τ_i writes the results of its computation to the register shared with τ_{i+1}, the interval $\delta_{i,i+1}$ between the instance release time $r_i \in A(\tau_i)$ of τ_i and the release time r_{i+1} of the instance of τ_{i+1} reading for the first time the data propagated by the instance of τ_i released at r_i is not larger than $\delta^{\mathsf{ub}}_{i,i+1}$ that is given by:*

- *If $P_i < P_{i+1}$ or $\pi(\tau_i) \neq \pi(\tau_{i+1})$,*

$$\delta^{\mathsf{ub}}_{i,i+1} = \begin{cases} \mathcal{R}_i + T_{i+1} - \eta_{i,i+1} & \mathsf{mod}(\mathcal{R}_i, \eta_{i,i+1}) = 0 \\ \mathcal{R}_i + T_{i+1} - \mathsf{mod}(\mathcal{R}_i, \eta_{i,i+1}) & \text{Otherwise} \end{cases} \quad (11)$$

- *If $P_i > P_{i+1}$ and $\pi(\tau_i) = \pi(\tau_{i+1})$,*

$$\delta^{\mathsf{ub}}_{i,i+1} = T_{i+1} - \eta_{i,i+1}. \quad (12)$$

Based on Theorems III.1 and III.2, we devise an efficient worst-case end-to-end latency analysis strategy for data chains of periodic real-time tasks executed by a partitioned fixed-priority preemptive scheduler upon a multiprocessor platform. Our algorithm aims to calculate relatively tighter, rather than exact, latency upper bounds with polynomial time complexity. To this end, based on Theorem III.1, we reduce the problem of bounding the worst-case latency of a data chain to a problem of bounding the releasing interval of data propagation instances for each adjacent pair of tasks in the data chain. Moreover, based on Theorem III.2, we derive a bound on the releasing interval of successive data propagation instances to yield the desired data chain latency bound. Owing to the problem reduction, we can efficiently compute the end-to-end latency upper bound by a simple accumulation.

Algorithm 1 Data chain worst-case latency bound analysis.

Require: A data chain F_n with $n \geq 1$ tasks.
Ensure: The worst-case latency bound $L^{\mathsf{ub}}(F_n)$ of data chain F_n.

```
1: if n == 1 then
2:     L^ub(F_n) ← T_1 + R_1
3:     return L^ub(F_n)
4: else
5:     L^ub(F_n) ← T_1 + R_n
6:     for i = 1 to n − 1 do
7:         η_{i,i+1} ← gcd(T_i, T_{i+1})
8:         if P_i > P_{i+1} and π(τ_i) = π(τ_{i+1}) then
9:             L^ub(F_n) ← L^ub(F_n) + T_{i+1} − η_{i,i+1}
10:        else
11:            if mod(R_i, η_{i,i+1}) == 0 then
12:                L^ub(F_n) ← L^ub(F_n) + R_i + T_{i+1} − η_{i,i+1}
13:            else
14:                L^ub(F_n) ← L^ub(F_n) + R_i + T_{i+1} − mod(R_i, η_{i,i+1})
15:    return L^ub(F_n)
```

Algorithm 1 shows the pseudocode of our data chain worst-case latency analysis method (named δ-Bound). If there is only one task in the data chain F_n, its worst-case latency bound is set to the sum of the period and the worst-case response time of its first task $\tau_1 = head(F_n)$ (Lines 1–3). Otherwise, the algorithm calculates the data chain worst-case latency bound by iteratively computing the bound of the releasing interval of data propagation instances for each pair of successive tasks in the data chain (Lines 5–15). By Theorems III.1 and III.2, we can conclude the correctness of our algorithm.

In this paper, we assume that the worst-case response times of all tasks in the system are known. The time complexity of Algorithm 1 depends on the number of tasks in the data chain and the time complexity of computing the greatest common divisor of periods of two tasks. The time complexity is $\mathcal{O}(\log T^{\mathsf{max}})$ to computing the greatest common divisor of periods of two tasks in the data chain with Euclidean algorithm, where T^{max} is the maximum period of the task in the chain. Therefore, the total time complexity of Algorithm 1 is $\mathcal{O}(n \log T^{\mathsf{max}})$, where n is the number of tasks in the chain.

IV. Experimental Evaluation

In this section, we experimentally compare the performance of our analysis method, δ-Bound, with the existing worst-case end-to-end latency analysis algorithms Davare-B from [11] and Kloda-B from [12]. We have two main goals for our evaluation: (1) to compare the performance of δ-Bound in terms of the analysis precision to that of Davare-B and Kloda-B; (2) to evaluate the analysis efficiency of δ-Bound.

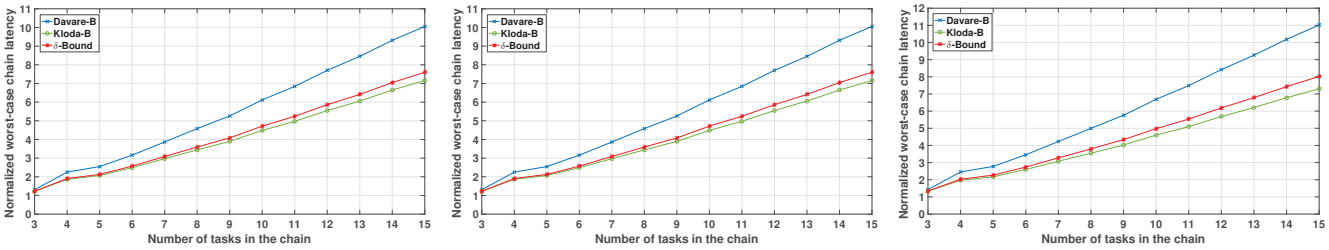
For our experiments, we used randomly generated task sets based on the automotive benchmark [14]. All tasks are periodic tasks with implicit deadlines, and they are scheduled under the rate monotonic (RM) scheduling policy. The period of a task is drawn at random from the set {1, 2, 5, 10, 20, 50, 100, 200, 1000} with an associated appearance probability given in [14]. As stated in [14], each generated chain is comprised of 3 different periods. The WCET C_i of each task τ_i in the task set is deduced based on its period T_i and utilization U_i generated by the UUniFast approach from [15]. We define the normalized utilization $\mathsf{U}_{\mathsf{nor}}(\Gamma)$ of a task set Γ to be:

$$\mathsf{U}_{\mathsf{nor}}(\Gamma) = \frac{\mathsf{U}_\Gamma}{m} \quad (13)$$

where U_Γ is the total utilization of task set Γ, and m is the number of processors.

Using the above parameters, we generated the task sets whose normalized utilization ranges from 0.25 to 0.75 with step size of 0.25, and discarded unschedulable task sets. For each of the presented data points, 10 000 schedulable task sets are examined. The worst-case response times are obtained from SimSo [16]. The experiments use an implementation in Python and are performed on a desktop computer with an Intel Core i7-8700 CPU @ 3.20GHz and 32GB of RAM.

Analysis Precision. Fig. 5 illustrates the average value of the calculated worst-case end-to-end latency versus the chain length for different normalized utilizations. Note that, the calculated end-to-end latencies are normalized with respect to the hyperperiod of the chain. We can find that δ-Bound *consistently outperforms* the existing worst-case latency analysis algorithm Davare-B. This is because, different from Davare-B, δ-Bound reduces the pessimism of the analysis result by taking



(a) $\mathsf{U}_{\mathsf{nor}}(\Gamma) = 0.25$ (b) $\mathsf{U}_{\mathsf{nor}}(\Gamma) = 0.50$ (c) $\mathsf{U}_{\mathsf{nor}}(\Gamma) = 0.75$

Fig. 5: Normalized worst-case chain latency for different algorithms in 4-processor systems.

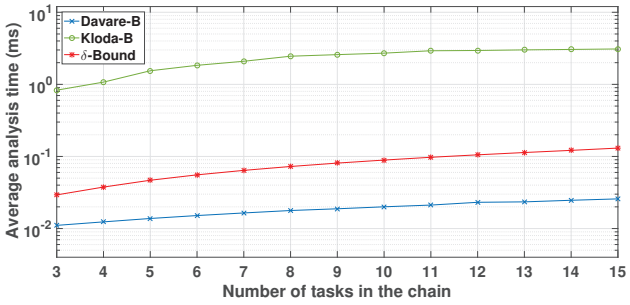


Fig. 6: Average analysis time in 4-processor systems for $\mathsf{U}_{\mathsf{nor}}(\Gamma) = 0.75$.

into account the overlapping of timing behavior of consecutive communication tasks in a chain during the latency bound analysis. The performance gap tends to widen as the chain length increases; the reason is that, as the chain length rises, more tasks will be included in a chain, and thus there are more opportunities for δ-Bound to improve the analysis precision by analyzing the coupled timing behavior of adjacent tasks in a chain. We can also see that Kloda-B delivers the most accurate estimates of the worst-case latency by enumerating all the possible releasing times of the first task in the chain within the hyperperiod. However, the gap between δ-Bound and Kloda-B is very small in most cases, revealing that, despite being sub-optimal, δ-Bound can provide a relatively tighter latency upper bound.

Analysis Efficiency. Fig. 6 shows the analysis time needed per task set in 4-processor systems for $\mathsf{U}_{\mathsf{nor}}(\Gamma) = 0.75$. Note that, a base 10 logarithmic scale is used for y-axis to make the result more clearly. Not surprisingly, the algorithm Davare-B with linear time complexity outperforms both δ-Bound and Kloda-B by separately analyzing the timing behavior of each task in a task chain. We also can see that δ-Bound achieves the similar analysis efficiency with Davare-B owing to the problem reduction, and its analysis time is much shorter than Kloda-B. For most task sets, the algorithm Kloda-B is dozens of times slower than δ-Bound. In summary, the efficiency improvement of δ-Bound against Kloda-B is significant.

V. Conclusions

We have proposed δ-Bound, an efficient latency bound analysis technique for data chains of periodic real-time tasks executed by a partitioned fixed-priority preemptive scheduler upon a multiprocessor platform. δ-Bound works by transforming the problem of bounding the worst-case latency of data chain to a problem of bounding the instance releasing interval for each pair of directly communicating tasks in the chain. Owing to such a problem reduction, we can effectively obtain a relatively tighter upper bound in polynomial time. Our evaluation shows δ-Bound not only achieves the similar analysis precision with the existing exponential time analysis algorithm, but also significantly improves analysis efficiency.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant No.: U1808206, 61602080, 61802185, 51978120, 71874020, 61761136019, and 61772112, the China Postdoctoral Science Foundation under Grant No.: 2018T110221 and 2016M591431, the Natural Science Foundation of Liaoning Province under Grant No.: 20180550540, the Natural Science Foundation of Jiangsu Province under Grant No.: BK20180470, the Social Science Foundation of Liaoning Province under Grant No.: L17CTQ002, and the Fundamental Research Funds for the Central Universities under Grant No.: DUT18RC(4)008 and 30919011233.

References

[1] AUTOSAR, "Specification of Timing Extensions (Version 4.3.1)," 2017.
[2] A. Hamann, D. Dasari *et al.*, "Communication centric design in complex automotive embedded systems," in *ECRTS*, 2017.
[3] J. Martinez, I. Sañudo, P. Burgio *et al.*, "End-to-end latency characterization of implicit and let communication models," in *WATERS*, 2017.
[4] J. Martinez, I. Sañudo *et al.*, "Analytical characterization of end-to-end communication delays with logical execution time," *IEEE TCAD*, 2018.
[5] ISO, "ISO 26262 Road vehicles - Functional safety," 2018.
[6] E. Mezzetti, L. Barbina *et al.*, "AURIX TC277 Multicore Contention Model Integration for Automotive Applications," in *DATE*, 2019.
[7] K. W. Tindell, A. Burns *et al.*, "An extendible approach for analyzing fixed priority hard real-time tasks," *Real-Time Systems*, 1994.
[8] M. Becker, D. Dasari *et al.*, "Synthesizing job-level dependencies for automotive multi-rate effect chains," in *RTCSA*, 2016.
[9] J. Schlatow and R. Ernst, "Response-time analysis for task chains in communicating threads," in *RTAS*, 2016.
[10] J. Schlatow and R. Ernst, "Response-time analysis for task chains with complex precedence and blocking relations," *ACM TECS*, 2017.
[11] A. Davare, Q. Zhu, M. Di Natale *et al.*, "Period optimization for hard real-time distributed automotive systems," in *DAC*, 2007, pp. 278–283.
[12] T. Kloda, A. Bertout, and Y. Sorel, "Latency analysis for data chains of real-time periodic tasks," in *ETFA*, 2018.
[13] M. Becker *et al.*, "End-to-end timing analysis of cause-effect chains in automotive embedded systems," *JSA*, 2017.
[14] S. Kramer, D. Ziegenbein, and A. Hamann, "Real world automotive benchmarks for free," in *WATERS*, 2015.
[15] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, 2005.
[16] M. Chéramy, P.-E. Hladik *et al.*, "SimSo: A Simulation Tool to Evaluate Real-Time Multiprocessor Scheduling Algorithms," in *WATERS*, 2014.