Analysis and Optimization of Worst-Case Time Disparity in Cause-Effect Chains

Xu Jiang[1], Xiantong Luo[1], Nan Guan[2*], Zheng Dong[3], Shaoshan Liu[4], Wang Yi[1,5]

[1] Northeastern University, China [2] City University of Hong Kong, China
[3] Wayne State University, USA [4]PerceptIn, USA [5]Uppsala University, Sweden

***Abstract*—In automotive systems, an important timing requirement is that the time disparity (the maximum difference among the timestamps of all raw data produced by sensors that an output originates from) must be bounded in a certain range, so that information from different sensors can be correctly synchronized and fused. In this paper, we study the problem of analyzing the worst-case time disparity in cause-effect chains. In particular, we present two bounds, where the first one assumes all chains are independent from each other and the second one takes the fork-join structures into consideration to perform more precise analysis. Moreover, we propose a solution to cut down the worst-case time disparity for a task by designing buffers with proper sizes. Experiments are conducted to show the correctness and effectiveness of both our analysis and optimization methods.**

***Index Terms*—automotive systems, cause-effect chain, sensor, timestamps, disparity**

I. Introduction

Automotive systems usually have deep processing pipelines that span over multiple stages processed by different software/hardware components with data dependencies. For example, in autonomous vehicles, obstacle avoidance is realized by a chain of computational tasks including sensing, perception, planning and control, as shown in Fig. 1[1]. The system must comply with timing constraints in several aspects to guarantee that the final control command outputs can be executed correctly and timely. Satisfying the end-to-end timing constraints of control paths is a prerequisite for correct and safe systems, e.g., the success of obstacle avoidance requires that the task chain finishes before predefined deadline. Violating timing constraints may lead to catastrophic consequences such as loss of human life. As a consequence, formal modeling and analysis must be performed to guarantee that the timing constraints are always honored at run-time. For this purpose, the end-to-end timing constraints have been extensively studied in the context of automotive systems [1]–[5], known as the so-called maximum reaction time, presenting the length of time interval from a stimulus to its corresponding response, and the maximum data age, describing the freshness of the data.

Except for the end-to-end latency, another important timing requirement in automotive system is that when some component receives data originated from different sensors, the time difference among the timestamps of the corresponding raw data (called time disparity in this work) must be in a certain range, so that information from different sensors can be synchronized and fused [6]. For example, the time difference between images and the LiDAR data used by the perception algorithm should be bounded by a certain threshold, so that the object perception can be correctly performed. This problem has also been proposed by autonomous driving industry [6] in RTSS 2021 Industry Challenge [6]. Unfortunately, little work has been presented for bounding the time disparity with theoretical guarantees.

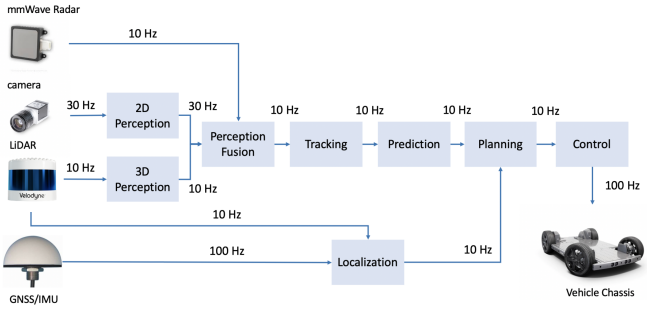


Fig. 1. A system example.

The system model studied in this paper is motivated by the recent industrial trend of allowing dynamic linking of services and clients during run-time such as AUTOSAR [7] initiative, known as cause-effect chains in literature. In this model, the system consists of multiple tasks executing on different processing units, and each task is statically assigned onto a processing unit. In particular, a task is activated periodically according to a given frequency and communicates with each other asynchronously.

In this paper, we develop analysis techniques to bound the worst-case time disparity of tasks in cause-effect chains. In particular, we present two upper-bounds of the worst-case time disparity of a task referring to the backward time of chains, where the first one assumes all chains are independent from each other and the second one takes the fork-join structures into consideration to perform more precise analysis. We also derive bounds for the worst-case and the best-case backward time of a chain by exploring the data propagating behaviors under non-preemptive scheduling algorithm, which is more precise than existing results. Moreover, we propose an optimization design to cut down the worst-case time disparity by designing buffers with different sizes. Experiments are conducted to verify our analysis techniques and optimization. The results show that our methods are effective and efficient.

*corresponding author: Nan Guan.

[1]The example is quoted from RTSS 2021 Industry Challenge, which is implemented by the company *PerceptIn* in autonomous driving systems.

978-3-9819263-7-8/DATE23/© 2023 EDAA

II. PRELIMINARY

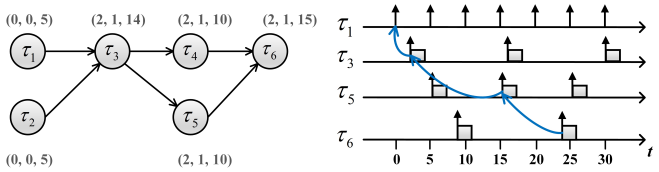
In this section, we present a formal characterization of the system model, which is motivated by the autonomous driving system developed by the company *PerceptIn* [6] and also complies with cause-effect chains in AUTOSAR compliant automotive systems [7].

A. System Model

We consider an application modeled as a Directed Acyclic Graph (DAG) G, called *cause-effect graph*, deployed on a hardware platform consisting of several Electronic Control Units (ECUs). The graph is denoted by $G = \langle V, E \rangle$, where V is the set of vertices and E the set of edges. Each vertex in V represents a task τ_i, and is characterized by a tuple $(W(\tau_i), B(\tau_i), T(\tau_i))$. $W(\tau_i)$ and $B(\tau_i)$ denote the Worst-Case Execution Time (WCET) and Best-Case Execution Time (BCET) of τ_i, respectively. $T(\tau_i)$ is the *period* of τ_i. An edge $(\tau_i, \tau_j) \in E$ denotes the input channel of τ_j (respectively, the output channel of τ_i), which can be considered as a buffer with size 1, and represents the data dependency between τ_i and τ_j. We call τ_i a *predecessor* of τ_j, and τ_j a *successor* of τ_i. A task with no incoming/outgoing edges is called a *source*/*sink* task of G. Each task is statically mapped to an ECU and the mapping is fixed in prior. The communicating between two tasks mapped to different ECUs is modeled as a periodic task on the bus [5], [8], [9], e.g., Control Area Networks (CANs) [10] .

We assume $W(\tau_i) = B(\tau_i) = 0$ if τ_i is a source task. This assumption is reasonable since source tasks are usually implemented as external stimuli that produce data without consuming any computing resource. Nevertheless, the assumption is made only for simplifying the definitions and notions in the paper, and does not limit the generality of our model.

A *cause-effect chain* (called chain for short) π is a path in G, i.e., a sequence of tasks $\pi = \{\pi^1, \pi^2, \cdots, \pi^{|\pi|}\}$ where π^i is a predecessor of π^{i+1} for each pair of consecutive elements π^i and π^{i+1} in π. The first and the last element in π are called the *head* and the *tail* task of π, respectively.



(a) A DAG. (b) A job chain.

Fig. 2. Examples.

An example of a cause-effect graph is shown in Fig. 2.(a), where the triple labeled next to each task τ_i corresponds to $(W(\tau_i), B(\tau_i), T(\tau_i))$. There are two source tasks in the graph: τ_1 and τ_2.

B. Run-Time Behavior

At run-time, each task releases an infinite sequence of *jobs* according to its period. We use J_i^k to denote the k^{th} job of task τ_i. $r(J_i^k)$, $s(J_i^k)$ and $f(J_i^k)$ denote the release time, start time and finish time of J_i^k, respectively. The first job of each task may be released with an arbitrary *release offset* relative to the start time of the entire system.

Tasks consume data tokens from its input channels (one or multiple) and produce output data tokens to their output channels. When a new data token arrives, the old data token is overwritten. That is, each job reads the latest available data token produced by its predecessors. When a job reads an input data token (or several input data tokens) and produces an output data token, the input data token is called a *cause* of the output data token. The data token produced by a source task indirectly being the cause of a data token is the *source* of the data token. For simplicity, we assume that the communication delay among jobs within the same ECU is zero. The reading and writing semantics complies with the *implicit* communication [11] implemented in automotive systems, e.g, AUTOSAR. More specifically, a job reads all data tokens from its input channels when it starts to execute, and writes the output data token to its output channels when it finishes. In particular, a source task produces output data tokens but does not read any data token. Each data token produced by a job J_i^k of a source task τ_i is attributed with a *timestamp* $t(J_i^k)$. Without loss of generality, we let $t(J_i^k) = r(J_i^k)$.

A non-preemptive fixed-priority scheduler is adopted for scheduling tasks assigned to the same ECU. We use $hp(\tau_i)$ to denote the set of tasks with higher priority than τ_i. The *worst-case response time* (WCRT) $\mathcal{R}(\tau_i)$ of τ_i is the longest length between the release time and finish time among all its jobs. τ_i is said to be *schedulable* if $\mathcal{R}(\tau_i) \leq T(\tau_i)$. The problem of bounding $\mathcal{R}(\tau_i)$ has been extensively studied in literature, and plentiful efficient analysis techniques have been presented [12], [13]. In this paper, we do not focus on the schedulability of the system, and simply assume that each task is schedulable.

C. Job Chain

A *job chain* of π is a sequence of jobs with data dependency, i.e., a job reads the data token associated with (not necessarily produced by) a job released by its predecessor in the chain for each pair of successive jobs. Numerous job chains can be constructed during run-time with respect to different scenarios.

The *immediate backward job chain*, which was first introduced by Dürr .el [5], is defined from the perspective of an output data token. The immediate backward job chain ending at the k^{th} job released by the tail task ($\pi^{|\pi|}$) of a chain π, denoted as $\overleftarrow{\pi_k}$, is defined iteratively from the last job in $\overleftarrow{\pi_k}$. We use $\overleftarrow{\pi_k}^i$ to denote the i^{th} job in $\overleftarrow{\pi_k}$.

Definition 1 (immediate backward job chain [5])**.** *For each $i = 1, \cdots, |\pi| - 1$, $\overleftarrow{\pi_k}^i$ is the last job of the i^{th} task in π that finishes its execution no later than the start time of $\overleftarrow{\pi_k}^{i+1}$.*

By definition, on each chain, the immediate backward job chain ending at the same job is unique. Under the implicit communication semantics, a job in $\overleftarrow{\pi_k}$ reads the data token produced by its predecessor in $\overleftarrow{\pi_k}$. Obviously, the output data token of the first job in $\overleftarrow{\pi_k}$ is the source of the output data token of the last job in $\overleftarrow{\pi_k}$. We define the *backward time* of $\overleftarrow{\pi_k}$ as $len(\overleftarrow{\pi_k}) = r(\overleftarrow{\pi_k}^{|\pi|}) - r(\overleftarrow{\pi_k}^1)$ [2]. In particular, we let

[2]The backward time is similar with the data age latency defined in many previous literature but a little different: the data age of the output produced by the k^{th} job of $\pi^{|\pi|}$ is defined as $f(\overleftarrow{\pi_k}^{|\pi|}) - r(\overleftarrow{\pi_k}^1)$.

$len(\overleftarrow{\pi_k}) = 0$ if the immediate backward job chain ending at the k^{th} job released by the tail task of π does not exist, e.g, the input channel is empty when some job in $\overleftarrow{\pi_k}$ starts to execute.

For example, the immediate backward job chain ending at a job released by τ_6 in Fig. 2.(a) on the chain $\{\tau_1, \tau_3, \tau_5, \tau_6\}$ is shown in Fig. 2.(b), where each curved arrow pointing from one job to another indicates two successive jobs in the job chain.

We use $\mathcal{W}(\pi)$ and $\mathcal{B}(\pi)$ to denote an upper bound and a lower bound of the *worst-case backward time* (WCBT) and the *best-case backward time* (BCBT) of π, respectively, i.e., $\mathcal{W}(\pi) \geq \max_{\forall k} len(\overleftarrow{\pi_k})$ and $\mathcal{B}(\pi) \leq \min_{\forall k} len(\overleftarrow{\pi_k})$.

III. Analysis

Definition 2 (Time Disparity)**.** *The time disparity of a job J_i^k, denoted by $\Delta(J_i^k)$ is the maximum difference among timestamps of all J_i^k's sources. The worst-case time disparity of τ_i is the maximum time disparity among all jobs of τ_i.*

The problem considered in this paper is to verify whether the time disparity of a task is bounded by a pre-defined value. For simplifying the presentation, in the following, we will drop the indices and use J to denote an arbitrary job of the analyzed task τ. We focus on bounding the time disparity of J, i.e, $\Delta(J)$. Obviously, each source of J can be traced through an immediate backward job chain from J to a job released by a source task. Let $\mathcal{P}$ denote the set of all chains where each starts from a source task and ends at τ. By definition:

$$\Delta(J) = \max_{\forall \lambda \neq \nu \in \mathcal{P}} |t(\overleftarrow{\lambda}^1) - t(\overleftarrow{\nu}^1)|$$

In the following, we focus on bounding $|t(\overleftarrow{\lambda}^1) - t(\overleftarrow{\nu}^1)|$, i.e., the difference of timestamps between the two sources that propagate to J through λ and ν. At this time, we assume that $\mathcal{W}(\pi)$ and $\mathcal{B}(\pi)$ of each chain π in $\mathcal{P}$ are known, and later we will introduce the computation for them. Without loss of generality, we let the release time of J be 0, i.e., $r(J) = 0$.

To better present the analysis, we call time interval $[a, b]$ a sampling window of $\overleftarrow{\pi}^1$ if $t(\overleftarrow{\pi}^1) \in [a, b]$. Intuitively, a sampling window of a source presents the time range of its timestamp. A straightforward result on the sampling window of $\overleftarrow{\pi}^1$ can be obtained by using $\mathcal{W}(\pi)$ and $\mathcal{B}(\pi)$.

Lemma 1. $-\mathcal{W}(\pi) \leq t(\overleftarrow{\pi}^1) \leq -\mathcal{B}(\pi)$.

Proof. By definition we know:

$$t(\overleftarrow{\pi}^1) = r(\overleftarrow{\pi}^1) = r(\overleftarrow{\pi}^{|\pi|}) - len(\overleftarrow{\pi}) = -len(\overleftarrow{\pi})$$

Since $-\mathcal{W}(\pi) \leq -len(\overleftarrow{\pi}) \leq -\mathcal{B}(\pi)$, the lemma holds. □

Theorem 1. *Let $\mathcal{O}_{\lambda,\nu} = \max\{|\mathcal{W}(\lambda) - \mathcal{B}(\nu)|, |\mathcal{W}(\nu) - \mathcal{B}(\lambda)|\}$, then it satisfies that:*

$$|t(\overleftarrow{\lambda}^1) - t(\overleftarrow{\nu}^1)| \leq \begin{cases} \mathcal{O}_{\lambda,\nu} & \lambda^1 \neq \nu^1 \\ \lfloor \frac{\mathcal{O}_{\lambda,\nu}}{T(\lambda^1)} \rfloor T(\lambda^1) & \lambda^1 = \nu^1 \end{cases}$$

Proof. When $\lambda^1 \neq \nu^1$, according to Lemma 1, $|t(\overleftarrow{\lambda}^1) - t(\overleftarrow{\nu}^1)| \leq \mathcal{O}_{\lambda,\nu}$ holds. When $\lambda^1 = \nu^1$, we know the difference of release times (timestamps) between two jobs of λ^1 must be multiple of its period $T(\lambda^1)$. Therefore, according to Lemma 1, the theorem is proved. □

Analogously with the second case in Theorem 1, the above result could be quite pessimistic if λ and ν have other common tasks except the head and tail tasks in them.

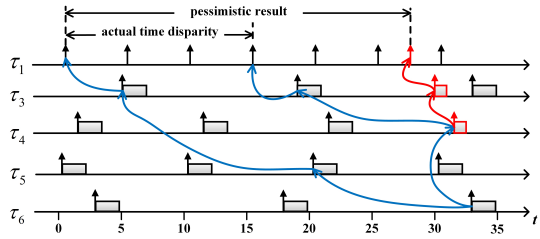


Fig. 3. Pessimism in Theorem 1.

Consider two chains $\lambda = \{\tau_1, \tau_3, \tau_5, \tau_6\}$ and $\nu = \{\tau_1, \tau_3, \tau_4, \tau_6\}$ in Fig. 2. When analyzing the upper bound of $|t(\overleftarrow{\lambda}^1) - t(\overleftarrow{\nu}^1)|$ for τ_6, the immediate backward job chains of J resulting in WCBT on λ and BCBT on ν are indicated by the curved arrows in Fig. 3 by considering these two chains to be independent of each other. However, this case never happens since the difference of release times between two jobs of τ_3 must be multiple of its period. This deviation will be magnified at each common task of these two chains when constructing immediate backward job chains on them to trace the sources of an output in the worst-case scenarios. Inspired by these observations, in the following, we derive a tighter bound by taking the fork-join structure into consideration.

Before going deep, we first introduce some results which will be useful in the following. Let J' be the k^{th} job released after/before J and it satisfies that $x \leq k \leq y$, where x and y are both integers (J' is released before J if k is negative). $\overleftarrow{\nu'}$ denotes the immediate backward job chain of J' on ν.

Lemma 2. $xT(\nu^{|\nu|}) - \mathcal{W}(\nu) \leq t(\overleftarrow{\nu'}^1) \leq yT(\nu^{|\nu|}) - \mathcal{B}(\nu)$.

Proof. By definition we know:

$$t(\overleftarrow{\nu'}^1) = r(\overleftarrow{\nu'}^1) = r(J') - len(\overleftarrow{\nu'}) = k \times T(\nu^{|\nu|}) - len(\overleftarrow{\nu'})$$

Since $x \leq k \leq y$, we know $xT(\nu^{|\nu|}) - len(\overleftarrow{\nu'}) \leq t(\overleftarrow{\nu'}^1) \leq yT(\nu^{|\nu|}) - len(\overleftarrow{\nu'})$. Moreover, since $\mathcal{B}(\nu) \leq len(\overleftarrow{\nu'}) \leq \mathcal{W}(\nu)$, it satisfies that: $xT(\nu^{|\nu|}) - \mathcal{W}(\nu) \leq t(\overleftarrow{\nu'}^1) \leq yT(\nu^{|\nu|}) - \mathcal{B}(\nu)$. The lemma is proved. □

Lemma 3. *It satisfies that:*

$$|t(\overleftarrow{\lambda}^1) - t(\overleftarrow{\nu'}^1)| \leq \begin{cases} \mathcal{O}^{x,y}_{\lambda,\nu} & \lambda^1 \neq \nu^1 \\ \lfloor \frac{\mathcal{O}^{x,y}_{\lambda,\nu}}{T(\lambda^1)} \rfloor T(\lambda^1) & \lambda^1 = \nu^1 \end{cases}$$

where $\mathcal{O}^{x,y}_{\lambda,\nu} = \max\{|\mathcal{W}(\nu) - \mathcal{B}(\lambda) - xT(\nu^{|\nu|})|, |\mathcal{B}(\nu) - \mathcal{W}(\lambda) - yT(\nu^{|\nu|})|\}$.

Proof. When $\lambda^1 \neq \nu^1$, $|t(\overleftarrow{\lambda}^1) - t(\overleftarrow{\nu'}^1)| \leq \mathcal{O}^{x,y}_{\lambda,\nu}$ holds by combining Lemma 1 and Lemma 2. When $\lambda^1 = \nu^1$, the difference of release times (timestamps) between two jobs of λ^1 must be multiple of its period $T(\lambda^1)$. Therefore, by combining Lemma 1 and Lemma 2, the lemma is proved. □

Now we are ready to derive a tighter bound of $|t(\overleftarrow{\lambda}^1) - t(\overleftarrow{\nu}^1)|$ when λ and ν have multiple common tasks. Let λ and ν have c tasks in common except the source tasks in them, denoted by $\{o_1, o_2, \cdots, o_c\}$, where o_i represents the i^{th} common task in λ and ν. Obviously, o_c is the analyzed task τ. We divide λ and ν into c sub-chains $\alpha_1, \alpha_2, \cdots, \alpha_c$ and $\beta_1, \beta_2, \cdots, \beta_c$, by following: 1) $\forall i \in [1, c]$, o_i is the tail task of both α_i and β_i, and 2) $\forall i \in [2, c]$, o_{i-1} is the head task of both α_i and β_i. For example, the chains $\{\tau_1, \tau_3, \tau_4, \tau_6\}$ and $\{\tau_2, \tau_3, \tau_5, \tau_6\}$ in Fig. 2 have two common tasks τ_3 and τ_6. When analyzing the maximum difference of timestamps between sources of τ_6's jobs on these two chains, we can divide them into sub-chains $\{\tau_1, \tau_3\}$, $\{\tau_3, \tau_4, \tau_6\}$ and $\{\tau_2, \tau_3\}$, $\{\tau_3, \tau_5, \tau_6\}$, respectively. Intuitively, each pair of α_i and β_i can be combined into a sub-graph with fork-join structure. By separating λ and ν into different pairs of sub-chains, we can derive a tighter bound of $|t(\overleftarrow{\lambda}^1) - t(\overleftarrow{\nu}^1)|$.

Theorem 2. *It satisfies that:*

$$|t(\overleftarrow{\lambda}^1) - t(\overleftarrow{\nu}^1)| \leq \begin{cases} \mathcal{O}^{x_1,y_1}_{\alpha_1,\beta_1} & \lambda^1 \neq \nu^1 \\ \lfloor \frac{\mathcal{O}^{x_1,y_1}_{\alpha_1,\beta_1}}{T(\lambda^1)} \rfloor T(\lambda^1) & \lambda^1 = \nu^1 \end{cases} \quad (1)$$

where x_1 and y_1 are computed recursively as follows:

$$\begin{aligned} j = c: \quad & x_j = 0, \quad y_j = 0 \\ j \in [1, c-1]: \quad & x_j = \lceil \frac{\mathcal{B}(\alpha_{j+1}) - \mathcal{W}(\beta_{j+1}) + x_{j+1}T(o_{j+1})}{T(o_j)} \rceil \\ & y_j = \lfloor \frac{\mathcal{W}(\alpha_{j+1}) - \mathcal{B}(\beta_{j+1}) + y_{j+1}T(o_{j+1})}{T(o_j)} \rfloor \end{aligned}$$

Proof. It is sufficient to prove the theorem by proving that the difference of release times between the jobs of o_1 in $\overleftarrow{\lambda}$ and $\overleftarrow{\nu}$ is in the range $[x_1T(o_1), y_1T(o_1)]$. If it is true, then according to Lemma 3, we know inequality (1) holds. In the next, we prove this by induction.

Suppose that the difference of release times between the jobs of o_j in $\overleftarrow{\lambda}$ and $\overleftarrow{\nu}$ is in the range $[x_jT(o_j), y_jT(o_j)]$. Clearly, it is true when $j = c$ since J is the job of o_j in both $\overleftarrow{\lambda}$ and $\overleftarrow{\nu}$. By considering the release time of the job of o_j in α_j as 0, according to Lemma 2, we know the release times of the jobs of o_{j-1} in $\overleftarrow{\lambda}$ and $\overleftarrow{\nu}$ are in the ranges $[-\mathcal{W}(\alpha_j), -\mathcal{B}(\alpha_j)]$ and $[x_jT(o_j) - \mathcal{W}(\beta_j), y_jT(o_j) - \mathcal{B}(\beta_j)]$, respectively. Therefore the difference of release times between the jobs of o_{j-1} in $\overleftarrow{\lambda}$ and $\overleftarrow{\nu}$ is in the range $T(o_{j-1})[\lceil \frac{\mathcal{B}(\alpha_j) - \mathcal{W}(\beta_j) + x_jT(o_j)}{T(o_{j-1})} \rceil, \lfloor \frac{\mathcal{W}(\alpha_j) - \mathcal{B}(\beta_j) + y_jT(o_j)}{T(o_{j-1})} \rfloor]$, i.e., $[x_{j-1}T(o_{j-1}), y_{j-1}T(o_{j-1})]$. By induction, we know the difference of release times between the jobs of o_1 in $\overleftarrow{\lambda}$ and $\overleftarrow{\nu}$ is in the range $[x_1T(o_1), y_1T(o_1)]$, and the theorem is proved. □

At last, by enumerating all combinations of chains in $\mathcal{P}$ and using Theorem 1 or 2, we can finally obtain the worst-case time disparity of the analyzed task τ. In particular, since the immediate backward job chain of a job on the same chain is unique, for each pair of chains in $\mathcal{P}$, we can consider the last joint task of them as the analyzed task for simplicity.

The end-to-end delay analysis in cause-effect chains has been extensively studied in literature, e.g., [1]–[4]. In the most recent work, Dürr .el [5] presented techniques for bounding the maximum data age of sporadic cause-effect chains, which can be directly applied to compute $\mathcal{B}(\pi)$ and $\mathcal{W}(\pi)$ for a chain π with a slight modification. However, they aimed to present a safe bound regardless of the applied scheduling algorithm. In the following, we focus on non-preemptive scheduling and present an upper bound of the backward time, which is more precise than the results presented in [5].

Lemma 4. *An upper bound of WCBT of π is given by $\mathcal{W}_\pi = \sum_{i=1}^{|\pi|-1} \theta_i$, where $\theta_i = T(\pi^i) + \mathcal{R}(\pi^i)$ if π^i and π^{i+1} are executed on different ECUs. Otherwise,*

$$\theta_i = \begin{cases} T(\pi^i) & \pi^i \in hp(\pi^{i+1}) \\ T(\pi^i) + \mathcal{R}(\pi^i) - (W(\pi^i) + B(\pi^{i+1})) & \pi^i \notin hp(\pi^{i+1}) \end{cases}$$

Proof. Considering an arbitrary job J of the last task in π, we construct the immediate backward chain $\overleftarrow{\pi}$ of J. By definition, $\overleftarrow{\pi}^{i-1}$ is the last job of π^{i-1} that finishes its execution no later than the start time of $\overleftarrow{\pi}^i$. When π^i and π^{i+1} are executed on different ECUs, we know $r(\overleftarrow{\pi}^{i+1}) - r(\overleftarrow{\pi}^i) \leq T(\pi^i) + \mathcal{R}(\pi^i)$. Otherwise, $\overleftarrow{\pi}^{i+1}$ cannot read the data token produced by $\overleftarrow{\pi}^i$, which must be overwritten by jobs of π^i released after $\overleftarrow{\pi}^i$ when $\overleftarrow{\pi}^{i+1}$ is released. In the next, we consider the case when π^i and π^{i+1} are executed on the same ECU. We prove the lemma by contradiction for each case.

- Suppose that $r(\overleftarrow{\pi}^{i+1}) - r(\overleftarrow{\pi}^i) > T(\pi^i)$ and $\pi^i \in hp(\pi^{i+1})$. Then we know the job of π^i released after $\overleftarrow{\pi}^i$ must have finished its execution before $\overleftarrow{\pi}^{i+1}$ since π^i has a higher priority than π^{i+1}. Reaching a contradiction.
- Suppose that $r(\overleftarrow{\pi}^{i+1}) - r(\overleftarrow{\pi}^i) > T(\pi^i) + \mathcal{R}(\pi^i) - (W(\pi^i) + B(\pi^{i+1}))$ and $\pi^i \notin hp(\pi^{i+1})$. Let t_s denote the start time of the job of π^i released after $\overleftarrow{\pi}^i$. Under non-preemptive scheduling, it must satisfy that $t_s \leq r(\overleftarrow{\pi}^i) + T(\pi^i) + \mathcal{R}(\pi^i) - W(\pi^i)$. On the other hand, $t_s > r(\overleftarrow{\pi}^{i+1})$ must hold, otherwise it must have finished its execution before $\overleftarrow{\pi}^{i+1}$ starts to execute due to the non-preemption. Since π^{i+1} has a higher priority than π^i, then we know $t_s > r(\overleftarrow{\pi}^{i+1}) + B(\pi^{i+1}) > r(\overleftarrow{\pi}^i) + T(\pi^i) + \mathcal{R}(\pi^i) - W(\pi^i)$. Reaching a contradiction.

In summary, we know $r(\overleftarrow{\pi}^{i+1}) - r(\overleftarrow{\pi}^i) \leq \theta_i$ holds in all cases. Therefore, we know $len(\overleftarrow{\pi}) = \sum_{i=1}^{|\pi|-1}(r(\overleftarrow{\pi}^{i+1}) - r(\overleftarrow{\pi}^i)) \leq \sum_{i=1}^{|\pi|-1} \theta_i$. The lemma is proved. □

In the next, we derive a lower bound of the BCBT of π.

Lemma 5. *A lower bound of the BCBT of π is given by $\mathcal{B}_\pi = \sum_{i=1}^{|\pi|} B(\pi^i) - \mathcal{R}(\pi^{|\pi|})$.*

Proof. Considering an arbitrary job J of the last task in π, we construct the immediate backward chain $\overleftarrow{\pi}$ of J. By definition, $\overleftarrow{\pi}^{i-1}$ is the last job of π^{i-1} that finishes its execution no later than the start time of $\overleftarrow{\pi}^i$, so we know, for each $i = 2, 3, \cdots, |\pi|$, it is satisfied that:

$$s(\overleftarrow{\pi}^i) - s(\overleftarrow{\pi}^{i-1}) \geq f(\overleftarrow{\pi}^{i-1}) - s(\overleftarrow{\pi}^{i-1}) \geq B(\pi^{i-1})$$

Then we know $s(\overleftarrow{\pi}^{|\pi|}) - s(\overleftarrow{\pi}^1) \geq \sum_{i=1}^{|\pi|-1} B(\pi^i)$. Moreover, since $f(\overleftarrow{\pi}^{|\pi|}) - s(\overleftarrow{\pi}^{|\pi|}) \geq B(\pi^{|\pi|})$, therefore $f(\overleftarrow{\pi}^{\pi}) -$

$s(\overleftarrow{\pi}^1) \geq \sum_{i=1}^{|\pi|} B(\pi^i)$ holds. Furthermore, since $f(\overleftarrow{\pi}^\pi) \leq \mathcal{R}(\pi^{|\pi|}) + r(\overleftarrow{\pi}^\pi)$ and $s(\overleftarrow{\pi}^1) \geq r(\overleftarrow{\pi}^1)$, we know

$$len(\overleftarrow{\pi}) = r(\overleftarrow{\pi}^\pi) - r(\overleftarrow{\pi}^1) \geq \sum_{i=1}^{|\pi|} B(\pi^i) - \mathcal{R}(\pi^{|\pi|})$$

The lemma is proved. □

Note that it is possible for the BCBT to be negative, indicating that the first job is released after the last job in an immediate backward job chain.

IV. Optimization

It can be observed from Lemma 3 that an output data token may originate from two sources produced by the same task. Apparently, this counter-intuitive result must be avoided. In this section, we discuss some possible solutions to cut down the worst-case time disparity of a task.

Some designing tricks are also widely used in many realistic implementations. For example, there are two typical choices for designating the period of τ_3 in the graph shown in Fig. 4: $30ms$ or $10ms$. Intuitively, when $T(\tau_3) = 10ms$, two-thirds of input data tokens of τ_3 may not propagate to the next task since τ_5's period is $30ms$. As a consequence, computation resources could be potentially wasted. However, one may raise the frequency of τ_3 to speed up the sampling for the data produced by τ_1, so that a smaller time disparity of τ_5 could be achieved. Counter-intuitively, this could be ineffective. As shown in Fig. 4, the execution pattern leading to the worst-case time disparity remains unchanged after τ_3's frequency is raised. This is because, the worst-case time disparity is largely decided by WCBT on one chain and BCBT on another, which may not change even when the sampling frequency is raised. This result can also be observed from Theorem 2. This frequency design is not artificial but extracted from the problem proposed by *PerceptIn* in the RTSS 2021 Industry challenge [6].

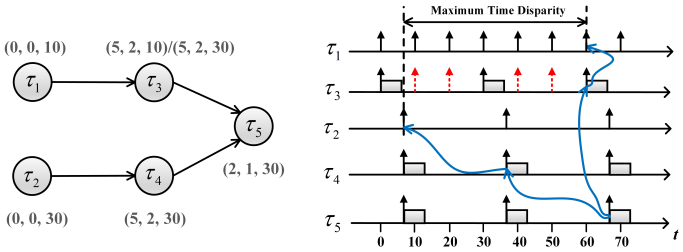


Fig. 4. An example of the frequency design.

In fact, it can be observed from Theorem 2 that the time disparity of a task with regard to two chains largely depends on the relative offset between the sampling windows of its sources, which should be as small as possible.

For this purpose, we propose to design a proper buffer size of the output channel of a source task so that its sampling window may be properly shifted to overlap with another one as much as possible. In the following, we first extend the result in Lemma 3 to the case where the buffer size of the input channel of a task is greater than 1, and then introduce how to decide a proper buffer size to cut down the worse-case time disparity.

A channel with a buffer size greater than 1 is organized in a FIFO manner. More specifically, each job always reads the first element in its input channel. When a new data token arrives, it is enqueued into the buffer, and the oldest data token, i.e., the first element, is removed from the buffer if the buffer is full. When a task has multiple successors, each output token produced by it is simultaneously written into the input channel of all its successors.

Algorithm 1: Design of the buffer size.

1 Given two chains λ and ν ending at the same task;
2 **for** *each* $i \in [1, c-1]$ **do**
3 Compute x_i and y_i using Theorem 2;
4 $A_\nu \leftarrow x_1 T(o_1) - \mathcal{W}(\beta_1)$; $B_\nu \leftarrow y_1 T(o_1) - \mathcal{B}(\beta_1)$;
5 $A_\lambda \leftarrow -\mathcal{W}(\alpha_1)$; $B_\lambda \leftarrow -\mathcal{B}(\alpha_1)$;
6 $M_\lambda = (A_\lambda + B_\lambda)/2$; $M_\nu = (A_\nu + B_\nu)/2$;
7 **if** *(*$M_\lambda \geq M_\nu$*)* **then**
8 Set the input buffer size of λ^2 to be $\lfloor \frac{M_\lambda - M_\nu}{T(\lambda^1)} \rfloor + 1$;
9 $L = \lfloor \frac{M_\lambda - M_\nu}{T(\lambda^1)} \rfloor T(\lambda^1)$;
10 **else**
11 Set the input buffer size of ν^2 to be $\lfloor \frac{M_\nu - M_\lambda}{T(\nu^1)} \rfloor + 1$;
12 $L = \lfloor \frac{M_\nu - M_\lambda}{T(\nu^1)} \rfloor T(\nu^1)$;
13 **return** L;

Lemma 6. *When the buffer size of the input channel of* π^2 *is* n *(*$n \geq 1$*), in the long term*[3]*, an upper bound* $\mathcal{W}(\pi)^n$ *of WCBT and a lower bound* $\mathcal{B}(\pi)^n$ *of BCBT of* π *are given by:* $\mathcal{W}(\pi)^n = \mathcal{W}(\pi) + (n-1)T(\pi^1)$ *and* $\mathcal{B}(\pi)^n = \mathcal{B}(\pi) + (n-1)T(\pi^1)$.

Proof. Due to space limitation, we omit the proof here. The intuition is that a task always reads the first element in the buffer whose timestamp is $(n-1)T(\pi^1)$ earlier than the newest arrived data. □

Inspired by the result in Lemma 6, we present an algorithm to decide proper buffer sizes referring to two chains, as shown in Algorithm 1. According to Lemma 2 and Theorem 2, we can obtain the sampling windows of the sources of J in two chains $\overleftarrow{\lambda}$ and $\overleftarrow{\nu}$, respectively, i.e., it satisfies that $t(\overleftarrow{\lambda}^1) \in [A_\lambda, B_\lambda]$ and $t(\overleftarrow{\nu}^1) \in [A_\nu, B_\nu]$ (line 4-6). Then the buffer size is decided by the relative offset between the midpoints of these two sampling windows (line 7-12). The intuition behind is to shift the sampling window on the right side to the left so that the offset between them is reduced.

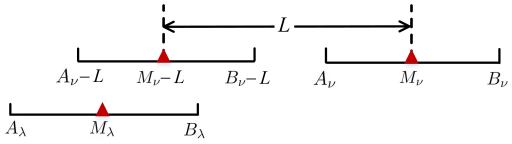


Fig. 5. Design principle of Algorithm 1.

Theorem 3. *It satisfies that:*

$$|t(\overleftarrow{\lambda}^1) - t(\overleftarrow{\nu}^1)| \leq \begin{cases} \mathcal{O}^{x_1,y_1}_{\alpha_1,\beta_1} - L & \lambda^1 \neq \nu^1 \\ \lfloor \frac{\mathcal{O}^{x_1,y_1}_{\alpha_1,\beta_1}}{T(\lambda^1)} \rfloor T(\lambda^1) - L & \lambda^1 = \nu^1 \end{cases} \quad (2)$$

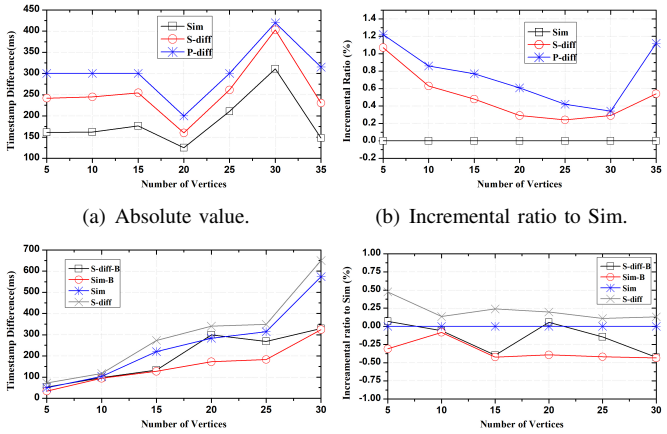
[3]Assume that the system has started for a while, and all buffers are full.

where $\mathcal{O}^{x_1,y_1}_{\alpha_1,\beta_1}$ is defined in Theorem 2, and L is computed by Algorithm 1.

Proof. The theorem is proved by combining Lemma 2 and Lemma 6. Due to space limitations, we omit the details here. The intuition is that the sampling window on the right side is shifted to the left since the buffer has a size greater than 1. So it becomes closer to the other one, as shown in Fig. 5. □

V. EVALUATION

In this section, we evaluate the effectiveness of our analysis techniques, as well as the optimization. Tasks are generated by using the synthesized automotive task sets presented by Kramer et.al [14] in WATERS challenge 2015. More specifically, tasks are generated in the following manner: 1) the periods of tasks are selected from the subset $\{1ms, 2ms, 5ms, 10ms, 20ms, 50ms, 100ms, 200ms\}$ according to their distributions in TABLE III of [14], and 2) the BCET and WCET of each task are computed by multiplying the ACET in TABLE IV of [14] with a related factor, which is uniformly picked from the range shown in TABLE V of [14].



(a) Absolute value. (b) Incremental ratio to Sim.

(c) Absolute value. (d) Incremental ratio to Sim.

Fig. 6. Evaluation of our methods.

In Fig. 6.(a) and (b), we compare the results obtained by Theorem 1 and Theorem 2, denoted as **P-diff** and **S-diff**, with the actual maximum time disparity obtained by simulation, denoted by **Sim**, which is a lower bound of the worst-case time disparity instead of a safe upper-bound. The cause-effect graph is generated by using the Python NetworkX function $dense\_gnm\_random\_graph$ [15]. Moreover, each graph is generated with a single sink task. The number of tasks in a graph is picked in the range $[5, 35]$ (X-axis). The release offset of each task τ_i is randomly picked from the range of $[1, T_i]$. Each graph is simulated for 10 times with different randomly generated offsets. For each configuration, we simulate for 10 minutes. At each point on X-axis, we generate 10 graphs and compute the average value of each result among all graphs. Fig. 6.(a) shows the absolute value of the time disparity under different methods, and Fig. 6.(b) shows the incremental ratio of different methods in comparison with **Sim**. It can be observed from Fig. 6.(a) that our analysis techniques are safe to bound the time disparity, and close to the lower bound. Moreover, **S-diff** performs much better than **P-diff** in the sense of estimation accuracy. From Fig. 6.(b), we can observe that the incremental ratio of **S-diff** is in general below 50%, indicating that our method can efficiently estimate the upper bound of time disparity in comparison with simulation results, which is not only unsafe but also time consuming.

In Fig. 6.(c) and (d), we compare the results obtained by Theorem 2 and the results obtained by simulation with their correspondences following Algorithm 1, denoted by **S-diff-B** and **Sim-B**, respectively. Since we only consider two chains in this case, the graph is generated by simply merging two independent chains at the same sink task, where the number of tasks on each chain is picked in the range $[5, 30]$ (X-axis). It can be observed that **S-diff-B** is much lower than **S-diff**, indicating that our design is effective for cutting down the worst-case time disparity. Most importantly, it can be observed that **Sim-B** is lower than **Sim**, indicating that our design could be quite useful in real implementation for cutting down the actual time disparity, not only the theoretical upper-bounds. Furthermore, our analysis results are quite close to the simulation results, and the incremental ratios are below 25% in most settings.

VI. CONCLUSION

In this paper, we present analysis techniques to bound the worst-case time disparity of a task in cause-effect chains. Moreover, we present an optimization to cut down the worst-case time disparity by designing buffers with proper sizes.

ACKNOWLEDGMENT

This work was partially supported by the National Natural Science Foundation of China (NSFC 62102072) and Research Grants Council of Hong Kong (GRF 11208522, 15206221).

REFERENCES

[1] M. Becker, D. Dasari, S. Mubeen, M. Behnam, and T. Nolte, "End-to-end timing analysis of cause-effect chains in automotive embedded systems," *JSA*, 2017.
[2] ——, "Analyzing end-to-end delays in automotive systems at various levels of timing information," *REACTION*, 2016.
[3] M. Günzel, K. Chen, N. Ueter, G. Brüggen, M. Dürr, and J. Chen, "Timing analysis of asynchronized distributed cause-effect chains," in *RTAS*, 2021.
[4] A. Kordon and N. Tang, "Evaluation of the age latency of a real-time communicating system using the let paradigm," in *ECRTS*, 2020.
[5] M. Dürr, G. V. D. Brüggen, K. Chen, and J. Chen, "End-to-end timing analysis of sporadic cause-effect chains in distributed systems," *TECS*, 2019.
[6] "http://2021.rtss.org/industry-session/," 2021.
[7] "https: //www.autosar.org/standards/adaptive-platform/," 2020.
[8] K. Lakshmanan, G. Bhatia, and R. Rajkumar, "Integrated end-to-end timing analysis of networked autosar-compliant systems," in *DATE*, 2010.
[9] Y. Zhao, V. Gala, and H. Zeng, "A unified framework for period and priority optimization in distributed hard real-time systems," *TCAD*, 2018.
[10] "Bosch. controller area network specification 2.0," 1991.
[11] A. Hamann, D. Dasari, S. Kramer, M. Pressler, and F. Wurst, "Communication centric design in complex automotive embedded systems," 2017.
[12] N. Guan, M. Stigge, W. Yi, and G. Yu, "New response time bounds for fixed priority multiprocessor scheduling," in *RTSS*, 2009.
[13] G. V. D. Bruggen, J. J. Chen, and W. H. Huang, "Schedulability and optimization analysis for non-preemptive static priority scheduling based on task utilization and blocking factors," in *RTS*, 2015.
[14] D. Z. S. Kramer and A. Hamann, "Real world automotive benchmark for free." *WATERS*, 2015.
[15] A. M. Kordon and N. Tang, "Evaluation of the Age Latency of a Real-Time Communicating System Using the LET Paradigm," in *ECRTS*, 2020.