Worst-Case Cause-Effect Reaction Latency in Systems with Non-Blocking Communication

Jakaria Abdullah*, Gaoyang Dai* and Wang Yi*
*Department of Information Technology
Uppsala University, Sweden

***Abstract*—In real-time embedded systems, a system functionality is often implemented using a data-flow chain over a set of communicating tasks. A critical non-functional requirement in such systems is to restrict the amount of time, i.e. cause-effect latency, for an input to impact its corresponding output. The problem of estimating the worst-case cause-effect latency is well-studied in the context of blocking inter-task communication. Recent research results show that non-blocking communication preserving functional semantics is critical for the model-based design of dynamically updatable systems. In this paper, we study the worst-case cause-effect reaction latency estimation problem in the context of non-blocking inter-task communication. We present a computationally efficient algorithm that tightly over-approximates the exact worst-case reaction latency in cause-effect data-flow chains.**

I. Introduction

A simple use case in real-time embedded applications is a data-flow chain where a sampler task samples an input, passes the sampled data to a controller task for processing and finally the processed output is used by an actuator task. The specification of the system often includes temporal constraints on such data-flow chains, also known as the end-to-end timing or latency constraints. More specifically, a latency constraint restricts the amount of time required before an input is taken into account by the corresponding output. Latency constraints are important temporal requirements in real-time systems implemented by multiple communicating tasks with different periods. Proving that the implementation of a system satisfies all such requirements is non-trivial but mandatory for safety-critical reasons.

Our work is motivated by a recent industrial trend of allowing dynamic linking of services and clients during run-time such as the Adaptive AUTOSAR [1] initiative. For run-time systems, a major challenge here is to quickly verify non-functional requirements such as the latency requirements as parts of a more complex resource optimization and mapping problem. However, a widely used system design practice is to verify all the functional and non-functional requirements using simulation before deployment. Such process is time-consuming as a simulation enumerates all possible data propagation paths to capture the worst-case situation. In the future, run-time systems may require to verify thousands of latency requirements during dynamic mapping [2] of an application where existing simulation-based latency estimation methods are clearly unsuitable.

In a recent work [3], it has been shown that non-blocking communication preserving functional semantics is critical for the design of dynamically updatable systems. The main reason behind it is that the blocking communication protocols used with priority-based scheduling are not designed to preserve any model-level functional semantics used in the system design tools [4]. In contrast, the Dynamic Buffering Protocol (DBP) [4] is a wait-free inter-task communication protocol that preserves a functional semantics similar to the one for synchronous programming [5] in implementation. However, latency estimation in data-flow chain using protocols like the DBP is limited to simulation-based approach [6].

In this context, we study the problem of estimating the reaction latency [7] of an input in a multi-rate data-flow chain of periodic tasks where tasks communicate using the DBP protocol. Our goal is to estimate the latency without simulation-like enumeration of data propagation paths so that the algorithm can be used during run-time. Here the main challenge lies in the multi-rate nature of the communication where a writer can write in a higher or a lower rate compared to its reader. As a result, an input may not propagate to output or may propagate multiple times. In this paper, we present an algorithm that tightly over-approximates the worst-case reaction latency of a periodic input in a data-flow chain without enumerating all data propagation paths. Our algorithm executes in linear time compared to the exponential-time algorithms of the simulation-based approach. We also present an experimental evaluation with a set of data-flow chains (typical representative of real-world systems) to show tightness of the values computed by our algorithm.

The rest of the paper is organized as follows. First in Section II, we give details of the problem and the system model considered in this work. Our proposed latency analysis method is described in Section III and evaluated in Section IV. In Section V, we review the previous related work. Finally, we conclude the paper with a summary together with future works in Section VI.

II. Problem Formulation

A. System Model

We assume a system S is implemented by a set of n periodic real-time tasks $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$. Each periodic task τ_i is characterized by a period T_i and a relative deadline $D_i \leq T_i$, both of which are positive integers. A job is an instance of a task, i.e one of the executions of a task. Each k^{th} job $J_{i,k}$ of

periodic task τ_i has a release time $r_i^k = r_i^{k-1} + T_i$, with $r_i^0 = 0, k \in \mathbb{N}$. Here by release time we refer to the time instant when job of a task becomes ready for execution. Each task τ_i is assigned a fixed task priority $\pi_i \in \mathbb{N}$ where a higher value means the task has higher priority. The system is scheduled using fixed priority preemptive scheduling. Finally, if for two tasks τ_i and $\tau_j (T_j \leq T_i)$, $T_i/T_j \in \mathbb{N}$, then their periods are harmonic.

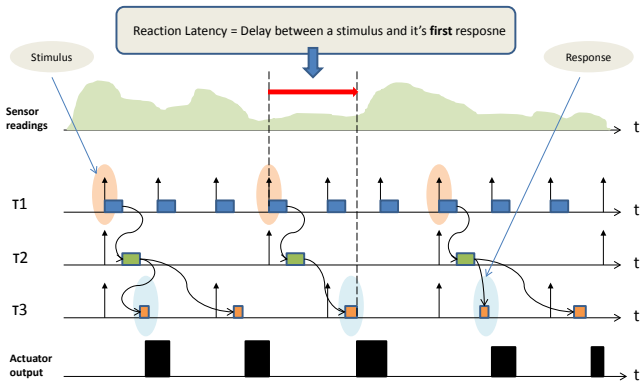

Fig. 1. Reaction latency in a cause-effect chain comprising three periodic tasks. The arrows indicate flow of data from one task to another one.

Definition 1 (Cause-effect Chain)**.** *We write $\tau_i \rightarrow \tau_j$ when there exists m, n, such that $J_{i,m} \rightarrow J_{j,n}$, which represents the fact that $J_{i,m}$ produces the data required for computing $J_{j,n}$. A cause-effect chain $C = \tau_1 \rightarrow \tau_2 \rightarrow \ldots \rightarrow \tau_N$ is a chain of $N \geq 2$ tasks where each of these except the first one (called stimulus) reads the data written by its predecessor and each of these except the last one (called response) writes the data for its successor (see Figure 1).*

Each of these cause-effect chains is associated with an end-to-end latency requirement. In this work, we are concerned with end-to-end latency from the perspective of a stimulus also known as the reaction latency. A reaction latency constraint of L_C time units to a particular chain C implies that any first response should occur no later than L_C time units after its corresponding stimulus [7]. Formally, we define reaction latency requirement over a cause-effect chain as follows:

Definition 2 (Reaction Latency Requirement)**.** *Given a cause-effect chain $C = \tau_1 \rightarrow \tau_2 \rightarrow \ldots \rightarrow \tau_N$ with $N \geq 2$ tasks we denote S_C as the set of job indexes of τ_1 that writes data that affects the jobs of τ_N. More specifically, an index $k \in S_C$, if there exists data dependency $J_{1,k} \rightarrow \ldots \rightarrow J_{N,l}$ for some l. Let $V_q(J_{p,k})$ denote the index of the first job of τ_q that depends on the data generated by the job $J_{p,k}$. If f_i^k is the finishing time of $J_{i,k}$ then the reaction latency requirement L_C on a cause-effect chain C is as follows:*

$$\forall k \in S_C, k \in \mathbb{N} : f_N^{V_N(J_{1,k})} - r_1^k \leq L_C. \tag{1}$$

As $\forall k, f_i^k \leq r_i^k + R_i$ where R_i is the worst-case response time (WCRT) [8] of τ_i, we define the worst-case reaction latency of C as:

$$W_C = \max_{k \in S_C} (r_N^{V_N(J_{1,k})} - r_1^k) + R_N. \tag{2}$$

B. Communication

The DBP protocol is designed for multi-task implementation of synchronous programs in uniprocessor. It defines the way a run-time system manages the shared memory or buffer used in inter-task communication. According to DBP, whenever a job is released, the run-time system modifies the pointer to the buffers that the released job will use for reading and writing. Similarly, when a reader job finishes, the run-time system marks the free buffers. For a simultaneous release of a reader and its writer, the pointer fixing operation for the writer is executed before the one for the reader. The protocol assumes each writer task has different priority than its reader tasks. If the system is schedulable, to ensure deterministic (in terms of value consumption) communication between a writer and its reader tasks, the DBP protocol has following rules:

- **High-to-low:** A low priority reader job reads the data written by the latest high priority writer job.
- **Low-to-high:** A high priority reader job reads the data written by the predecessor job of the latest low priority writer job.

Here a latest writer job means the last job that is released before or at the same time with the reader job under consideration.

To use the DBP protocol in a multi-core platform, the only new requirement is that the tasks communicating from different cores should have low-to-high priority communication. This requirement is necessary as the high-to-low communication in DBP assumes the reader can not start before finishing time of its writer.

C. Problem Statement

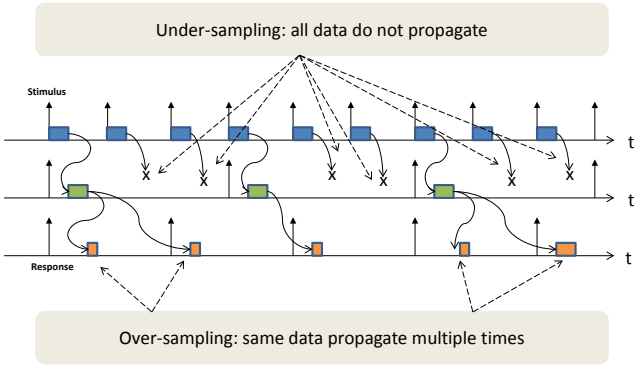

Fig. 2. The effect of over-sampling and under-sampling in multi-rate cause-effect chain.

A cause-effect chain may consist of tasks executing at different rates or periods. Such multi-rate data-flow chains thus often suffer from effects of under-sampling or over-sampling of data. Under-sampling happens when a slow reader misses

some data written by a fast writer. Over-sampling occurs when a fast reader reads the same data from a slow writer multiple times. With these effects, it is challenging to calculate the reaction latency of a multi-rate cause effect chain due to the following reasons (see Figure 2):

- If the chain contains any under-sampling effect then the calculation should only consider the input that reaches the output.
- If the chain contains any over-sampling effect then the calculation should only consider delay of the first reader job (out of multiple readers that read the same data).

In this context, we study the following problem in this paper: Given a cause-effect chain $C = \tau_1 \rightarrow \tau_2 \rightarrow \ldots \rightarrow \tau_N$ of $N \geq 2$ synchronously released periodic tasks executing under fixed priority preemptive scheduling and communicating using the DBP communication protocol, we want to calculate W_C without enumerating all possible data propagation paths of the chain.

III. Reaction Latency Estimation

A. Chains with Two Tasks

In the simplest case, we first consider a cause-effect chain $C = \tau_i \rightarrow \tau_o$ of only two synchronously released periodic tasks. Let $gcd(T_i, T_o)$ to be the greatest common divisor of the periods T_i and T_o. We have the following lemma about distances between release times of two communicating tasks.

Lemma 1. *Let T_i and T_o are the periods of two synchronous periodic tasks τ_i and τ_o, $gcd(T_i, T_o)$ is the minimum distance between the release time of a job of τ_o and the release time of the nearest job of τ_i that is released after it.*

Proof. Let d be the distance between release times of writer job $J_{i,m}$ and reader job $J_{o,n}$ where $m \in \mathbb{Z}_{>0}$ and $n \in \mathbb{Z}_{>0}$ are job indexes. We assume $J_{o,n}$ is released before $J_{i,m}$ so

$$d = m \cdot T_i - n \cdot T_o. \tag{3}$$

We want to find the smallest value of $d \in \mathbb{Z}_{>0}$.

First we assume T_i and T_o are co-prime and $gcd(T_i, T_o) = 1$. Let $n_1 \in \mathbb{Z}_{>0}, n_2 \in \mathbb{Z}_{>0}$ are co-prime and $N = n_1 \cdot n_2$. Then according to Chinese Remainder Theorem (CRT) [9], if a_1 and a_2 are integers such that $0 \leq a_i < n_i$ for every $i \in \{1, 2\}$, then there is one and only one integer x, such that $0 \leq x < N$ and the remainder of the Euclidean division of x by n_i is a_i for every $i \in \{1, 2\}$.

In our setting, $N = T_i \cdot T_o$ is the hyper-period of the tasks τ_i and τ_o. According to CRT, there exists unique t such that $0 \leq t < N$ for every pair (a_1, a_2) with $a_1 \in \{0, 1, \ldots, T_i - 1\}$ and $a_2 \in \{0, 1, \ldots, T_o - 1\}$ where

$$\begin{aligned} t &\equiv a_1 \mod T_i \\ t &\equiv a_2 \mod T_o. \end{aligned} \tag{4}$$

So there exists $m \in \mathbb{Z}_{>0}$ and $n \in \mathbb{Z}_{>0}$ for which

$$\begin{aligned} & m \cdot T_i + a_1 = n \cdot T_o + a_2 \\ \implies & m \cdot T_i - n \cdot T_o = a_2 - a_1 = d. \end{aligned} \tag{5}$$

If we assume $a_1 = 0$, then $a_2 = 1$ is the smallest $a_2 \in \mathbb{Z}_{>0}$ for which $0 \leq t < N$ exists. So the smallest value of $d \in \mathbb{Z}_{>0} = a_2 = 1 = gcd(T_i, T_o)$.

Next we consider the case where T_i and T_o are not co-prime, $gcd(T_i, T_o) \neq 1$. As $gcd(T_i, T_o)$ is the greatest common divisor of T_i and T_o then $T_o/gcd(T_i, T_o)$ and $T_i/gcd(T_i, T_o)$ are co-prime. In this case $N = (T_i \cdot T_o)/(gcd(T_i, T_o))^2 = lcm(T_i, T_o)/gcd(T_i, T_o)$ where $lcm(T_i, T_o)$ is the least common multiple of T_i and T_o. Note that $lcm(T_i, T_o)$ also represents the hyper-period of these tasks which is scaled here by $gcd(T_i, T_o)$. Now we have from the previous case,

$$\begin{aligned} & m \cdot T_i/gcd(T_i, T_o) - n \cdot T_o/gcd(T_i, T_o) = 1 \\ \implies & m \cdot T_i - n \cdot T_o = gcd(T_i, T_o). \end{aligned} \tag{6}$$

Combining both cases, we get the smallest value of $d \in \mathbb{Z}_{>0} = gcd(T_i, T_o)$. ∎

Now to compute release distances between two communicating jobs, we have to consider high-to-low and low-to-high communication where the DBP protocol uses different operations based on the priorities of the tasks. The maximum delay between release times of a writer job and its first reader job is given by the following theorem:

Theorem 1. *In periodic task chain $\tau_i \rightarrow \tau_o$, for any $J_{i,p} \rightarrow J_{o,q}$ where $q = V_o(J_{i,p})$, the maximum distance between release times r_i^p and r_o^q using the DBP protocol is*

$$\Delta_{i,o} = \begin{cases} T_i + \min(T_i, T_o) - gcd(T_i, T_o), & \text{if } \pi_i < \pi_o \\ \min(T_i, T_o) - gcd(T_i, T_o), & \text{if } \pi_i > \pi_o \end{cases} \tag{7}$$

where $min(T_i, T_o)$ is the minimum of the periods T_i and T_o.

Proof. First we consider $\pi_i < \pi_o$ or low to high priority communication. There are three cases based on the periods of the communicating tasks. As r_i^{p+1} is the release time of the latest writer job with respect to r_o^q, $r_o^q - r_i^p$ will be maximum if $r_o^q - r_i^{p+1}$ is maximized. In the first case where T_i and T_o are harmonic (including the case $T_i = T_o$), the distance $r_o^q - r_i^p$ is always T_i. The reason behind it is that in harmonic case the release time of a job of a task with larger period will always coincide with the release time of one of the jobs of task with shorter period, as a result $r_o^q - r_i^{p+1}$ is always zero. In the next two cases, we consider T_i and T_o to be non-harmonic. In the case where $T_i < T_o$, if $r_o^q \neq r_i^{p+1}$ then $r_i^{p+1} < r_o^q < r_i^{p+2}$. We rewrite $r_o^q - r_i^{p+1}$ as $(r_i^{p+2} - r_i^{p+1}) - (r_i^{p+2} - r_o^q) = T_i - (r_i^{p+2} - r_o^q)$ which maximizes for the minimum value of $(r_i^{p+2} - r_o^q) = gcd(T_i, T_o)$. So the maximum value of $r_o^q - r_i^p$ is $T_i + T_i - gcd(T_i, T_o)$. For the case $T_i > T_o$, if $r_o^q \neq r_i^{p+1}$ then $r_o^{q-1} < r_i^{p+1} < r_o^q$. We rewrite $r_o^q - r_i^{p+1}$ as $(r_o^q - r_o^{q-1}) - (r_i^{p+1} - r_o^{q-1}) = T_o - (r_i^{p+1} - r_o^{q-1})$. As the minimum value of $(r_i^{p+1} - r_o^{q-1})$ is $gcd(T_i, T_o)$ (by Lemma 1), the maximum value of $r_o^q - r_i^p$ in this case is $T_i + T_o - gcd(T_i, T_o)$. Combining these three cases, we have the equation for $\pi_i < \pi_o$.

Next we consider $\pi_i > \pi_o$ or high to low priority communication. Now r_i^p is the release time of the latest writer job that

propagates data to the reader job released at r_o^q. When T_i and T_o are harmonic (including the case $T_i = T_o$), the distance $r_o^q - r_i^p$ is always zero as the release time of a reader job always coincides with the release time of the writer job that writes data for it. For non-harmonic T_i and T_o, we have two cases. First when $T_i < T_o$ and $r_o^q \neq r_i^p$, then $r_i^p < r_o^q < r_i^{p+1}$. We rewrite $r_o^q - r_i^p$ as $(r_i^{p+1} - r_i^p) - (r_i^{p+1} - r_o^q)$. So, $r_o^q - r_i^p$ will be maximum for the minimum value of $r_i^{p+1} - r_o^q = gcd(T_i, T_o)$. So, the maximum value of $r_o^q - r_i^p$ is $(r_i^{p+1} - r_i^p) - (r_i^{p+1} - r_o^q) = T_i - gcd(T_i, T_o)$. In case of $T_i > T_o$ and $r_o^q \neq r_i^p$, we have $r_o^{q-1} < r_i^p < r_o^q$. We rewrite $r_o^q - r_i^p$ as $(r_o^q - r_o^{q-1}) - (r_i^p - r_o^{q-1})$. Here $r_o^q - r_i^p$ will be maximum for minimum value of $r_i^p - r_o^{q-1} = gcd(T_i, T_o)$. So, the maximum value of $r_o^q - r_i^p$ is $(r_o^q - r_o^{q-1}) - (r_i^p - r_o^{q-1}) = T_o - gcd(T_i, T_o)$. Combining these cases, we get the equation for maximum delay in high-to-low communication. ■

Now using equations 2 and 7, for $C = \tau_i \rightarrow \tau_o$ we get

$$W_C = \Delta_{i,o} + R_N. \tag{8}$$

B. Chains with $N > 2$ Tasks

In this section we extend the worst-case latency computation results for two task chain to cause-effect chain $C = \tau_1 \rightarrow \tau_2 \rightarrow \ldots \rightarrow \tau_N$ with $N > 2$ periodic tasks. A straightforward approach would be to add maximum release distances for each two task segment $\tau_i \rightarrow \tau_o$ of C to calculate W_C as $\sum_{i=1}^{N-1} \Delta_{i,i+1} + R_N$. However, this approach gives an unsafe estimation which we will explain next.

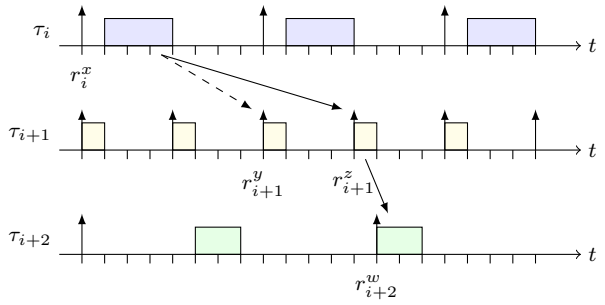

Fig. 3. An example showing increased latency due to the situation when the first read data is not propagated. The arrows indicate data-flow between tasks. The dashed arrow shows the first read.

To show the problem of only adding the local worst-cases in latency computation, let $\tau_i \rightarrow \tau_{i+1} \rightarrow \tau_{i+2}$ be a segment of a cause-effect chain $C = \tau_1 \rightarrow \tau_2 \rightarrow \ldots \rightarrow \tau_N$ with $N > 2$ periodic tasks. Let $J_{i,x} \rightarrow J_{i+1,y}$ where $y = V_{i+1}(J_{i,x})$ and $J_{i+1,z} \rightarrow J_{i+2,w}$ where $w = V_{i+2}(J_{i+1,z})$. If both $J_{i+1,y}$ and $J_{i+1,z}$ read from the same job of τ_i and $w = V_{i+2}(J_{i,x})$, we have a case where the first job of a reader task that reads a specific data from its writer task is not propagating it to the next task of the chain but the data eventually propagates. An example of such situation is shown on Figure 3 where $\pi_{i+2} < \pi_i < \pi_{i+1}$ and $T_{i+1} < T_i < T_{i+2}$. This shows that any worst-case reaction latency computation algorithm is unsafe if it does not consider the delay due to over-sampling and under-sampling of data.

An intuition to calculate a safe upper bound on the worst-case reaction latency is like this, if a task reads the same data multiple times then we should consider the number of its outputs that can be missed by its reader in the chain. Given $\tau_i \rightarrow \tau_{i+1} \rightarrow \tau_{i+2}$, we denote H_i^{i+2} as the maximum possible distance between release times of two jobs of τ_2 that are the first reader job of a data and the job that first propagates it respectively. Now to calculate H_i^{i+2}, we present two theorems about maximum number of over-sampling and under-sampling of data due to multi-rate communication.

Theorem 2. *In a cause-effect chain $\tau_i \rightarrow \tau_o$ with $T_i < T_o$, the maximum number of data writes M_i^o that a job of reader task τ_o can miss from its writer task τ_i using the DBP protocol is*

$$M_i^o = \left\lceil \frac{T_o}{T_i} \right\rceil - 1. \tag{9}$$

Proof. Let $j > 0$ and $k > j$ be the indexes of the first job and the last job in the longest sequence of $(k - j + 1) \geq 2$ writer jobs of τ_i whose output can be missed by the reader jobs of τ_o. We also assume $J_{i,j-1} \rightarrow J_{o,m}$ and $J_{i,k+1} \rightarrow J_{o,m+1}$. We have two cases based on priorities of τ_i and τ_o.

Case high to low: In this case, as the reader always reads from the latest writer job, $r_i^{j-1} \leq r_o^m < r_i^j$ and $r_i^{k+1} \leq r_o^{m+1} < r_i^{k+2}$. So $(k+1) \cdot T_i - j \cdot T_i < T_o \implies (k-j+1) < \frac{T_o}{T_i}$. From this we see the maximum possible integer value of $(k - j + 1)$ is $\lceil \frac{T_o}{T_i} \rceil - 1$.

Case low to high: In this case, the reader always reads from the predecessor of the latest writer job. As j is the index of the first missed job then $r_i^j \leq r_o^m < r_i^{j+1}$. Similarly, as k is the index of the last missed job then $r_i^{k+2} \leq r_o^{m+1} < r_i^{k+3}$. These give the condition $(k+2) \cdot T_i - (j+1) \cdot T_i < T_o$. From this we get the maximum possible integer value of $(k-j+1)$ as $\lceil \frac{T_o}{T_i} \rceil - 1$.

Finally, we consider the special case where the length of the longest sequence of missed jobs is one. If $k > 0$ is the index of the missed job then we have $r_i^{k+1} - r_i^k < T_o < r_i^{k+1} - r_i^{k-1} \implies 1 < \frac{T_o}{T_i} < 2$. For $1 < \frac{T_o}{T_i} < 2$, $\lceil \frac{T_o}{T_i} \rceil = 2$, so the Equation 9 also satisfies for this case. ■

Theorem 3. *In a cause-effect chain $\tau_i \rightarrow \tau_o$ with $T_i > T_o$, the maximum number of reader jobs of τ_o that reads data from the same writer job of τ_i using the DBP protocol is*

$$U_i^o = \left\lceil \frac{T_i}{T_o} \right\rceil. \tag{10}$$

Proof. Let $j \geq 0$ and $k > j$ be the indexes of the first job and the last job in the longest sequence of $(k - j + 1) \geq 2$ reader jobs of τ_o that reads from the same job $J_{i,m}$ of τ_i. Now we consider two cases based on DBP rules.

Case high to low: In this case, $r_o^j \geq r_i^m$ and $r_o^k < r_i^{m+1}$. As a result, $k \cdot T_o - j \cdot T_o < T_i \implies (k-j) < \frac{T_i}{T_o}$. From this we see the maximum possible integer value of $(k-j)$ is $\lceil \frac{T_i}{T_o} \rceil - 1$. So maximum number of missed jobs is $(k - j + 1) = \lceil \frac{T_i}{T_o} \rceil$.

Combination	H_i^{i+2}
$T_i > T_{i+1}, T_{i+1} \geq T_{i+2}$	0
$T_i > T_{i+1}, T_{i+1} < T_{i+2}$	$\min(M_{i+1}^{i+2}, U_i^{i+1} - 1) \cdot T_{i+1}$
$T_i \leq T_{i+1}, T_{i+1} \geq T_{i+2}$	0
$T_i \leq T_{i+1}, T_{i+1} < T_{i+2}$	$M_{i+1}^{i+2} \cdot T_{i+1}$

Fig. 4. Effect of different combinations of over-sampling and under-sampling in H_i^{i+2} of $\tau_i \rightarrow \tau_{i+1} \rightarrow \tau_{i+2}$

Case low to high: As j is the index of first job that reads from $J_{i,m}$, $r_o^j \geq r_i^{m+1}$. Similarly, as k is the index of last job that reads from $J_{i,m}$ then $r_o^k < r_i^{m+2}$. These give the condition $k \cdot T_o - j \cdot T_o < (m+2-m-1) \cdot T_i$. From this we get the maximum possible integer value of $(k-j+1)$ as $\lceil \frac{T_i}{T_o} \rceil$. ■

Now we observe that, for $T_{i+1} \geq T_{i+2}$, τ_{i+2} can never under-sample τ_{i+1} and $H_i^{i+2} = 0$. If $T_{i+1} < T_{i+2}$, then there are two cases. In the first case, $T_i \leq T_{i+1}$ and under-sampling τ_{i+2} may miss maximum M_{i+1}^{i+2} jobs of τ_{i+1}. However, if $T_i > T_{i+1}$ then out of maximum M_{i+1}^{i+2} jobs of τ_{i+1} that a job of τ_{i+2} may miss, maximum $U_i^{i+1} - 1$ of these are propagating the same data read from τ_i. We summarize all possible values of H_i^{i+2} in the table of Figure 4.

Finally we can calculate the worst-case latency of a chain with $N > 2$ tasks as

$$W_C^\star = \sum_{i=1}^{N-1} \Delta_{i,i+1} + \sum_{i=1}^{N-2} H_i^{i+2} + R_N. \tag{11}$$

We denote the above mentioned algorithm to calculate reaction latency as *Algorithm Safe-Latency (SL)*. Now we present a proof to show that *Algorithm SL* computes safe over-approximation of W_C.

Theorem 4. *For cause-effect chains with only periodic tasks, algorithm SL calculates a safe over-approximation of the worst-case reaction latency.*

Proof. Given a chain $C = \tau_1 \rightarrow \tau_2 \rightarrow \ldots \rightarrow \tau_N$ with $N > 2$ periodic tasks, let $\gamma = J_{1,a1} \rightarrow J_{2,a2} \rightarrow \ldots \rightarrow J_{N,aN}$ be the data-flow path between jobs responsible for the worst-case reaction latency. Now for any three job segment $J_{i,p} \rightarrow J_{o,q} \rightarrow J_{l,s}$ that is part of γ, we have to show that *Algorithm SL* over-approximates $r_l^s - r_i^p$. We have two cases. First if $q \in V_o(J_{i,p})$, then by equation 7, $r_s^l - r_i^p = r_s^l - r_o^q + r_o^q - r_i^p \leq \Delta_{i,o} + \Delta_{o,l}$. Secondly if $q \notin V_o(J_{i,p})$, then multiple jobs of τ_o reads from the same $J_{i,p}$. Let $J_{o,f} \in V_o(J_{i,p})$ where $f < q$. Now according to definition of H_i^{i+2}, the distance $r_o^q - r_o^f$ can be maximum H_i^l. So, in both cases *Algorithm SL* safely over-approximates the release distance between jobs that propagates the data. ■

IV. Evaluation

For evaluation, we consider periodic tasks from the Bosch case study of an EMS [10]. The case study is for a multi-core platform with a global memory and local scratchpads. We ignore memory access overhead and the only effect of placing tasks in different cores in our analysis is that the periodic tasks doing inter-core communication should have low to high priority communication. The given mapping of tasks is such, core 2: tasks with periods $2, 5, 20, 50, 100, 200, 1000$ milliseconds (ms), core 1: 1 ms task, core 3: 10 ms task. All the tasks are assigned unique priorities which is compatible with the DBP protocol. Figure 5 presents the cause-effect chains that we considered for experiment. Unfortunately, the case study provides only one representative chain (C1 in Figure 5). We considered two more possible chains C2 and C3 keeping the same mapping and priority information provided in the case study. Additionally, we used three more chains C4, C5 and C6 to show the effect of the non-harmonic periods.

Chain	Periods	Priority Order	LCM
C1	$100, 10, 2$	$[7, 11, 13]$	100
C2	$50, 2, 5, 1$	$[8, 13, 12, 15]$	50
C3	$20, 50, 10, 100, 1$	$[9, 8, 11, 7, 15]$	100
C4	$100, 10, 50, 60$	$[7, 11, 13, 2]$	300
C5	$50, 15, 40, 30$	$[4, 2, 3, 1]$	600
C6	$40, 15, 100, 30$	$[1, 5, 3, 6]$	600

Fig. 5. Different cause-effect chains used in our experiment.

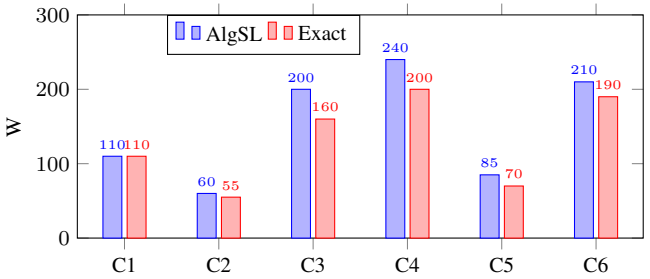

Fig. 6. Comparison of reaction latencies computed by our algorithm and the exact approach for chains of Figure 5.

We implemented the *Algorithm SL* described in Section III using Python programming language. To compare, we compute the exact maximum delay between releases of a periodic stimulus and its corresponding response by checking all possible data-flow paths within the hyper-period of a chain. We call this computation as *Exact* and it has a time-complexity linear in the length of the hyper-period of the tasks in a chain. The length of hyper-period is in the worst-case exponential with respect to the largest period of a task in the chain. In contrast, *Algorithm SL* has complexity which is linear in the number of tasks in a chain. As we see in Figure 6, our algorithm is able to over-approximate the exact value in a tight way and for the chains with only harmonic periods the computed value is almost exact. We also notice that our over-approximation becomes large for chains with extreme under-sampling (like 1 ms to 100 ms communication) which is a rare case for periodic task chain implementation in real-world systems.

V. Related Work

Our focus in this paper is on data propagation between independently triggered tasks, in contrast to the state-of-the-art end-to-end response time computation [11] considering the

first response at the end of a chain of tasks, where tasks may trigger each other. The analysis of latency constraints in multi-rate systems using asynchronous (non-blocking) communication was first studied in the context of synthesizing task parameters [12]. A renewed interest on such analysis stems from an industrial publication [13] where the authors propose a framework to calculate end-to-end latencies in automotive systems supporting the asynchronous communication model. However, latency estimation algorithms given in [13] are dependent on response times of the intermediate jobs and do not consider functional semantics preservation. Existing state-of-the-art frameworks and tools [14] for latency computation are simulation-based and only suitable for design-time use. In contrast, we consider latency estimation without path simulation in a setting where model-level functional semantics can be preserved using additional buffer places and a non-blocking communication protocol.

An alternative non-blocking communicaiton concept called the logical execution time (LET) [15] uses time-triggered communication at the release time (read) and the deadline (write) of the task. In [16], it has been shown that although LET is suitable for composable software, it increases delay in data reading as a task can finish computation of data long before its deadline. In contrast, the DBP protocol allows low priority readers to read data before the deadline of its high priority writers. Finally, our algorithm can be used for reaction latency estimation in LET by only allowing low-to-high communication.

Prelude [6] is a data-flow language intended for the design of multi-rate control systems preserving functional semantics of synchronous programming [5]. It allows simulation-based latency computation [17] using model-level data dependencies. We note that it is possible to implement the DBP protocol in Prelude. In that sense, our algorithm allows latency computation of Prelude programs during run-time.

VI. Conclusion

In this paper, we have studied the problem of estimating the worst-case reaction latency of a periodic input in a multi-rate cause-effect chain that uses non-blocking inter-task communication. We provide a computationally efficient algorithm which computes a safe and tight over-approximation of the exact worst-case reaction latency of chains with both harmonic or non-harmonic periods. Our algorithm does not depend on WCRTs of the data writer jobs and only assumes the system to be schedulable. An evaluation based on [10] shows, in presence of write-read pair with only harmonic periods, our algorithm computes the same value as the exact one. In case of communication between tasks with non-harmonic periods, our algorithm provides tight over-approximations as seen in Figure 6. Overall, low computation cost and tightness of computed values make our algorithm suitable to be used by a run-time system.

As future work, we want to evaluate benefits of our algorithm with any optimization algorithm that is used during dynamic mapping of applications. The algorithm can be improved further to handle chains with sporadic input and communication overhead related to a platform.

References

[1] "Autosar adaptive platfrom," autosar.org. [Online]. Available: https://www.autosar.org/standards/adaptive-platform/

[2] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: Survey of current and emerging trends," in *Proceedings of the 50th Annual Design Automation Conference*, ser. DAC '13, 2013, pp. 1:1–1:10.

[3] W. Yi, "Towards customizable cps: Composability, efficiency and predictability," in *Formal Methods and Software Engineering*, Z. Duan and L. Ong, Eds., 2017, pp. 3–15.

[4] P. Caspi, N. Scaife, C. Sofronis, and S. Tripakis, "Semantics-preserving multitask implmentation of synchronous programs," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 2, pp. 15:1–15:40, 2008.

[5] A. Benveniste, P. Caspi, S. A. Edwards, N. Halbwachs, P. L. Guernic, and R. de Simone, "The synchronous languages 12 years later," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 64–83, 2003.

[6] C. Pagetti, J. Forget, F. Boniol, M. Cordovilla, and D. Lesens, "Multi-task implementation of multi-periodic synchronous programs," *Discrete Event Dynamic Systems*, vol. 21, no. 3, pp. 307–338, Sep. 2011.

[7] "Specification of timing extensions," autosar.org. [Online]. Available: https://www.autosar.org/fileadmin/user_upload/standards/classic/4-0/AUTOSAR_TPS_TimingExtensions.pdf

[8] M. Joseph and P. Pandya, "Finding response times in a real-time system," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.

[9] D. E. Knuth, *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Boston, MA, USA: Addison-Wesley, 1997.

[10] S. Kramer, D. Ziegenbein, and A. Hamann, "Real world automotive benchmark for free," in *6th Workshop on Tools and Methodologies for Embedded and Real-time Systems at ECRTS 15*, Jul 2015.

[11] J. Schlatow and R. Ernst, "Response-time analysis for task chains in communicating threads," in *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2016, pp. 1–10.

[12] R. Gerber, S. Hong, and M. Saksena, "Guaranteeing end-to-end timing constraints by calibrating intermediate processes," in *Proceedings of Real-Time Systems Symposium*, 1994, pp. 192–203.

[13] N. Feiertag, K. Richter, J. Nordlander, and J. Jonsson, "A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics," in *Workshop on Compositional Theory and Technology for Real-Time Embedded Systems*, 2008.

[14] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the symta/s approach," *IEE Proceedings - Computers and Digital Techniques*, vol. 152, no. 2, pp. 148–166, 2005.

[15] T. A. Henzinger, B. Horowitz, and C. M. Kirsch, "Giotto: a time-triggered language for embedded programming," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 84–99, 2003.

[16] S. Matic and T. A. Henzinger, "Trading end-to-end latency for composability," in *26th IEEE International Real-Time Systems Symposium (RTSS'05)*, 2005, pp. 12 pp.–110.

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