Improving the Reaction Latency Analysis of Message Synchronization in ROS

Chenhao Wu*, Ruoxiang Li†, Naijun Zhan‡*, and Nan Guan†
*SKLCS, Institution of Software, Chinese Academy of Sciences, Beijing, China
†City University of Hong Kong, Hong Kong SAR
‡School of Computer Science, Peking University, Beijing, China
{wuch, znj}@ios.ac.cn, nanguan@cityu.edu.hk, ruoxiang.li@my.cityu.edu.hk

***Abstract*—Multi-sensor data fusion plays a crucial role in modern autonomous systems, enabling them to perceive the surroundings from multiple dimensions. However, the accuracy of fusion output can be compromised by the temporal inconsistency of input messages from different sources. ROS provides algorithms of message synchronization to mitigate this misalignment before the data fusion process. Nevertheless, this introduces additional latency in processing each message, which influences the real-time performance of ROS systems. Previous research [1] is the first to analyze and bound the reaction latency of the *ApproximateTime* synchronization policy in ROS, which is essential for analyzing the system-level end-to-end reaction time. However, their bound is overly pessimistic. In this paper, we propose a safe and tight reaction latency upper bound for the *ApproximateTime* policy. We conduct experiments to validate its accuracy and assess its improvements compared to [1].**

I. Introduction

Modern autonomous systems rely on real-time data from multi-sensors to obtain multi-dimensional perception of their surroundings. Data fusion is pivotal in integrating information and reducing noise. However, messages from different sources often exhibit significant disparities in sampling time. This discrepancy poses challenges for the practical application of data fusion. Without synchronized message input, the accuracy and reliability of fusion outcomes are greatly diminished. Therefore, synchronizing the sampling times of data from different sensors before the fusion process is crucial.

In the widely-used Robot Operating System (ROS), message filters are provided as software components to synchronize messages. Extensive research has been conducted on the *ApproximateTime* synchronization policy, a standard synchronization algorithm in ROS that is widely employed in various autonomous systems. Studies [2], [3] have demonstrated its excellent performance in minimizing the sampling time disparity of messages.

While the synchronization policy in ROS effectively mitigates the disparity of sampling time, it also introduce additional latency in processing each message. To ensure the real-time performance of ROS, it is important to quantify and upper bound the end-to-end latency it contributes. Recent research [1] is the first work analyzing the latency caused by the *ApproximateTime* synchronization policy. An upper bound is given on the reaction latency, which measures the maximal time delay caused by synchronization as a part of the system-level end-to-end reaction time.

However, their reaction latency upper bound is overly pessimistic, making it difficult to be used in end-to-end real time analysis or real-world development. In this paper, we propose a more accurate reaction latency upper bound that is both safe and tight. We conduct experiments under different settings to verify that our bound performs better in all cases.

II. Background and Related Works

A. Background

ROS [4] is a set of open-source software libraries and tools for developing robot applications and autonomous systems. In the latest version *Iron Irwini* [5] of ROS 2, four standard message synchronization policies are provided.

- The *ExactTime* Policy [6]: It selects the output messages with exactly the same timestamp. Its output sets are perfectly aligned ideal for data fusion. However, it is too restrictive for real-world systems and is seldom used.
- The *ApproximateEpsilonTime* Policy [7]: It selects message sets whose messages' maximal timestamp difference is no larger than a user-predefined threshold ϵ, as opposed to 0 in the *ExactTime* Policy. However, the optimal value of ϵ is difficult to be determined or even does not exist.
- The *LatestTime* Policy [8]: It uses statistics to publish at the highest frequency among sensors. As it prioritizes the output frequency, the temporal consistency of its outputs may be relatively high.
- The *ApproximateTime* Policy [9]: It predicts the incoming messages and selects the output set from all the available messages with the minimum timestamp difference. This enables the policy to adapt to different input patterns and output the best-aligned messages possible. Nonetheless, [10] points out that it focuses on the short-term optimization and may yield worse long-term results.

B. Related Works

Data fusion algorithms often assume that messages are perfectly aligned in terms of sampling time, which is not the case in the real world. Techniques [11], [12] have been developed to tackle this issue. But they only work below an upper bound of the temporal inconsistency.

ROS 2 provides real-time performance enhancements based on ROS. Researches have been conducted on the response time analysis of ROS 2. [13] models the execution of ROS

2 applications as processing chains and performs an end-to-end reaction time analysis. [14] proposes an end-to-end timing analysis for cause-effect chains in ROS2, considering the maximum end-to-end reaction time and maximum data age. However, all these works ignore the latency caused by the message synchronizer.

Recent work [2] models the *ApproximateTime* Policy and analyzes its time disparity upper bound. [3] investigates other real-time properties of this policy. [10] proposes a novel policy and illustrates its optimality regarding time disparity. The only work analyzes the latency caused by message synchronizer is [1], which provides upper bounds on the passing latency and reaction latency. In this paper, we aim to propose a better reaction latency upper bound compared to [1].

III. Problem Definition

A. System Model

The system contains N sensors, each sporadically sampling messages. We use m_i^j to denote the j-th message generated by the i-th sensor. We may omit the superscript j when it is clear or irrelevant in the context. The timestamp of m_i^j, indicating when it is sampled by the sensor, is denoted as $\tau(m_i^j)$. The timestamp difference between consecutive messages of the i-th sensor is bounded within $[T_i^L, T_i^U]$, where $0 < T_i^L \leq T_i^U$. Note that the case of periodicity $T_i^L = T_i^U$ is considered.

After sampled by sensors, messages are transmitted via corresponding queue to the *Synchronizer*. We denote the queue of sensor i as Q_i. We assume the transmission is reliable and queues are sufficiently long, ensuring no message loss or overflow. The delay caused by the transmission or other pre-processing is bounded within $[D_i^L, D_i^U]$ for messages sampled by sensor i. It holds that $0 \leq D_i^L \leq D_i^U$, which encompasses the scenario of ignoring the delay $D_i^U = 0$. We denote the time when message m_i^j arrives at the *Synchronizer* as $\alpha(m_i^j)$. We assume that the delay will not disrupt the sampling order of messages, i.e., $\tau(m_i^j) < \tau(m_k^l) \implies \alpha(m_i^j) < \alpha(m_k^l)$.

The *Synchronizer* selects output messages based on synchronization policy. In this paper we focus on the timing behaviors of the *ApproximateTime* synchronization policy in ROS, which will be elaborated in section V.

B. Problem Definition

In this paper, we focus on the reaction latency, whose definition is given below:

Definition 1. ***(Reaction latency)*** *[1]. Let m_i^k be a published message, t_f denote the time when m_i^k is published, and m_i^j be the latest published message in $\{m_i^l \mid l < k\}$. The reaction latency experienced by m_i^k, denoted as $R(m_i^k)$, is defined as the difference between the arrival time of m_i^j and the published time of m_i^k, i.e., $R(m_i^k) = t_f - \alpha(m_i^j)$.*

Fig. 1 shows an example of reaction latency of m_1^2, which is the time interval between the arrival of m_1^1 and m_1^2's published time. Intuitively, the reaction latency of a specific sensor represents the maximum delay introduced by *Synchronizer* as a part of the end-to-end reaction time, which measures the time the whole system needs to react to an external event. In a worst-case scenario, an external event happens just after m_1^1 is sampled (yellow lightning). So the first sensor does not detect it until m_1^2 is sampled, and the system can know about it until its publishing.

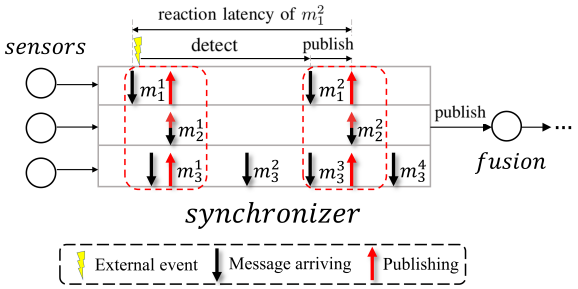

Fig. 1. Example of reaction latency

We assume the following system information:

- N: the total number of the sensors;
- T_i^L and T_i^U: the lower and upper bound of timestamp difference between consecutive messages from sensor i;
- D_i^L and D_i^U: the lower and upper bound of delay respecting messages from sensor i.

Based on these parameters, we aim to derive a safe and tight upper bound for reaction latency corresponding to any published message of the *ApproximateTime* Policy.

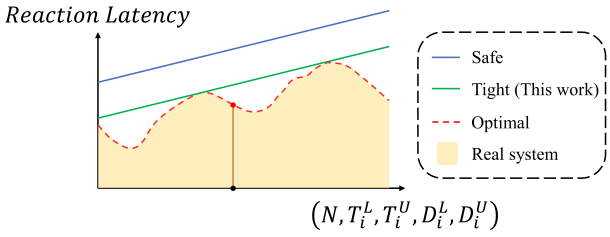

Fig. 2. Illustration of bound properties

Clearly, for any valuation to the set of parameters (e.g. the black point at x-axis in Fig. 2), there may be infinitely many real systems sharing the same values to their parameters, but with different reaction latency, i.e., the black point corresponds to the orange segment in yellow area. The top point of the segment (red dot), referred to as optimal valuation bound, represents the largest reaction latency achievable for a given parameter valuation. Any value no less than optimal bound is called a valuation bound. Now, we define a parameter bound which associates a valuation bound to each valuation of the parameters. We say a parameter bound is *safe* if any associated valuation bound is no less than the corresponding optimal valuation bound. Also, we say a parameter bound is *tight* if there exists some associated valuation bound exactly same as the corresponding optimal valuation bound. Fig. 2 illustrates the concepts *safe*, *tight* and *optimal*.

In what follows, we will abuse parameter bound as bound.

IV. The *ApproximateTime* Policy

In this section, we introduce the model of the *ApproximateTime* Policy presented in [2]. This model abstracts away details irrelevant to the reaction latency and serves as a basis for our further analysis. We first introduce some basic concepts.

A. Concepts

The time disparity of a message set is defined as the maximal difference of the timestamps of any two messages in the set.

Definition 2 (Time disparity [2])**.** *Let $S = \{m_1, \cdots, m_N\}$ be a set of messages. The time disparity of S, denoted by $\Delta(S)$, is defined as the difference between the largest and smallest timestamps of messages in S, i.e., $\Delta(S) = \max_{m_i \in S} \tau(m_i) - \min_{m_j \in S} \tau(m_j)$.*

We define the pivot as the message with the largest timestamp among the first messages of each queue. It represents the message that must be chosen into the next output set and serves as a reference point for the policy to choose other messages.

Definition 3 (Pivot [2])**.** *Let $S = \{m_1, \cdots, m_N\}$, where each m_i is the first message in Q_i. The pivot $m_{\mathbf{P}}$ is the message with the largest timestamp in S. When multiple messages in S with the largest timestamp, the pivot will be the message with the maximum queue index.*

We call a message set S regular *if it contains N elements from N different queues.*

Definition 4 (Selected Set [2])**.** *Let $m_{\mathbf{P}}$ be a pivot and Γ be the set of all regular sets that include $m_{\mathbf{P}}$. The selected set is the set with the smallest time disparity in Γ. When multiple sets in Γ with the smallest time disparity, the selected set $S = \{m_1, ..., m_N\}$ satisfies the condition: for any regular set $S' = \{m'_1, ..., m'_N\} \in \Gamma$, if $\Delta(S') = \Delta(S)$ then $\forall i, \tau(m_i) \leq \tau(m'_i)$.*

Each queue Q_i stores all the arrived messages along with a special message as the tail, called the predicted message. The predicted message represents the optimistic prediction of the incoming message, and its timestamp is set to the last arrived message's timestamp plus T_i^L.

B. Model

The *ApproximateTime* Policy aims to publish the message sets with the smallest time disparity. This is achieved by identifying the pivot and choosing the selected set. However, if not all queues have sufficient arrived messages, predicted messages may be included in the selected set. In such case, the system opts to wait for the corresponding message to arrive.

Algorithm 1 outlines the *ApproximateTime* Policy. When a new message arrives, the system replaces the old predicted message in Q_i and generates a new one (Line 1-3). The pivot and selected set is calculated based on their definitions (Line 5 - 7). If the selected set does not contain any predicted message, it will be published (Line 9). All messages in the selected set and all messages preceding them will be discarded (Line 10-11). If the selected set contains predicted messages, the system will wait for these messages to arrive (Line 13).

Algorithm 1: The *ApproximateTime* Policy [2]

Input: the newly arrived message m_i
1 discard the old predicted message in Q_i;
2 put m_i to the end of Q_i;
3 generate a new predicted message with timestamp $\tau(m_i) + T_i^L$ and put it to the end of Q_i;
4 **while** *each queue has at least one* arrived *message* **do**
5 $m_{\mathbf{P}} \leftarrow$ the current pivot (Definition 3);
6 **if** *all predicted messages' timestamps* $> \tau(m_{\mathbf{P}})$ **then**
7 $S \leftarrow$ the selected set (Definition 4);
8 **if** *all messages in S are arrived messages* **then**
9 publish S;
10 **for** *each* $m_j \in S$ **do**
11 discard m_j and all messages before m_j in the corresponding Q_j;
12 **else**
13 **return**;
14 **else**
15 **return**;
16 **return**;

C. An Illustrative Example

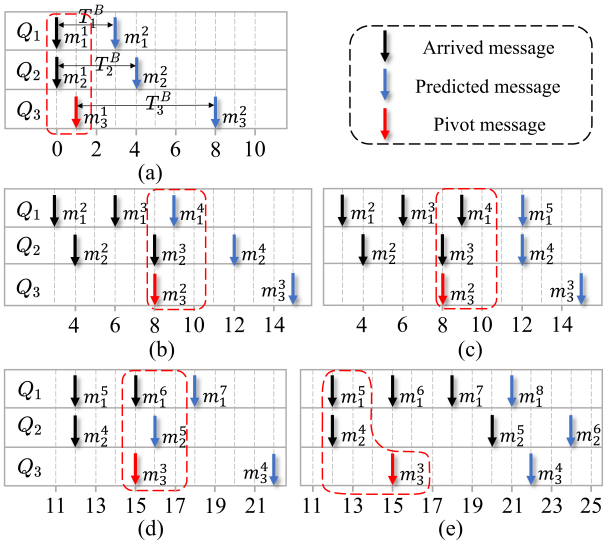

Fig. 3. An example of the *ApproximateTime* Policy

Fig. 3 illustrates an example of the *ApproximateTime* Policy, showcasing in the sequence (a) - (e). The x-axis represents the timestamp. In this example, there are 3 sensors inputting messages through Q_1 to Q_3. The timestamp difference bounds of each sensor is $T_1^L = T_1^U = 3$, $T_2^L = 4$, $T_2^U = 7$, and $T_3^L = T_3^U = 6$. Since the *ApproximateTime* Policy selects outputs based solely on messages' timestamp, the delay after sampling will not affect the composition of the published messages. For simplicity we ignore the delay in this example, so the black arrow represents both the timestamp and the arrival time of each message.

Fig. 1-(a) depicts the initial state of the system. The first messages of queue 1 and 2 arrive at time 0, while m_3^1 arrives at time 1. Consequently, by time 1, each queue has at least one message and the latest first message, m_3^1 (highlighted in red), becomes the pivot. In this scenario, the predicted message of each queue is distant from the pivot, and the selected set comprises only the arrived messages (circled with a red box). According to Algorithm 1, the system will publish the selected set and discard the messages.

Fig. 3-(b) and (c) depict a scenario of waiting for predicted message. After the publishing and discarding process shown in (a), messages of each queue arrive after their respective T^L. At time 8 in (b), m_3^2 arrives and becomes the new pivot. Since m_1^4, which is the predicted message of the first queue, is closer to the pivot than the arrived message m_1^3, it is included in the selected set. According to Algorithm 1, the system chooses to wait for this predicted message. When m_1^4 actually arrives at time 9 in (c), The system selects the same selected set, which now contains only arrived messages. So the system publishes.

Fig. 3-(d) and (e) illustrate the scenario where message is later than predicted. After (c), all messages continue to arrive at the minimal timestamp interval. At time 15 in (d), m_3^3 arrives and becomes the pivot. The system chooses a selected set including a predicted message m_2^5 and opts to wait for it. In (e), m_2^5 finally arrives after the maximal timestamp interval $T_2^U = 7$ at time 20, which is later than predicted. Now m_2^5 is a worse choice compared to m_2^4. Consequently, the system chooses and publishes a new selected set $\{m_1^5, m_2^4, m_3^3\}$, which is different from predicted.

V. Reaction Latency Upper Bound

In [1], a safe reaction latency upper bound of the *ApproximateTime* Policy is derived. However, this bound is pessimistic. In the following, we will derive a new reaction latency upper bound which is safe and tight. A worst-case system is given as the proof of tightness.

A. New Upper Bound

Consider the reaction latency of m_i^k. Use m_i^j to denote the preceding published message in the same queue, t_f^k to denote the publishing time of m_i^k. Denote the published sets containing m_i^j (resp. m_i^k) by $S^1_{\textbf{PUB}}$ (by $S^2_{\textbf{PUB}}$). Recall $R(m_i^k) = t_f^k - \alpha(m_i^j)$. In [1], the reaction latency is divided into two parts: $R(m_i^k) = (t_f^k - \alpha(m_i^k)) + (\alpha(m_i^k) - \alpha(m_i^i))$. Upper bounds of the two parts are calculated separately, and the reaction latency upper bound is given by summing them up. However, for the first part, the worst case requires m_i^k to be the earliest in $S^2_{\textbf{PUB}}$. For the second part, the worst case requires m_i^k to be the latest in $S^2_{\textbf{PUB}}$. This inconsistency results in the pessimism of their reaction latency upper bound.

By contrast, we propose to divide the reaction latency into three parts to avoid inconsistency. Assume the pivot of $S^2_{\textbf{PUB}}$ is $m^2_{\textbf{P}}$ in queue p, i.e., $m^2_{\textbf{P}} = m_p^2$. Use m_p^1 to denote the preceding message of m_p^2. The timestamps of m_p^1 and m_p^2 are our dividing points. We first prove that m_p^1 is in $S^1_{\textbf{PUB}}$.

Lemma 1. *Let $S^1_{\textbf{PUB}}$ and $S^2_{\textbf{PUB}}$ be two consecutive published sets, where $S^2_{\textbf{PUB}}$ is later than $S^1_{\textbf{PUB}}$. Let m_p^2 represent the pivot of $S^2_{\textbf{PUB}}$ and m_p^1 denoting its preceding message. m_p^1 must be included in $S^1_{\textbf{PUB}}$.*

Proof. According to the definition of the pivot (Definition 3), there is no message in Q_p before m_p^2 when the system is going to choose and publish $S^2_{\textbf{PUB}}$. This implies that m_p^1 is discarded.

According to Algorithm 1, the system only discards all the published messages and those preceding them (Line 11). Since m_p^1 is the message preceding m_p^2 and m_p^2 is not yet published at this time, m_p^1 must be discarded as a published message.

Given that $S^1_{\textbf{PUB}}$ is the previous published set, so m_p^1 is included in $S^1_{\textbf{PUB}}$. □

Now we know the two separation points m_p^1 and m_p^2 are included in the two published sets respectively. The reaction latency of m_i^k is thus divided into three parts:

- the latency from $\alpha(m_i^j)$ to $\tau(m_p^1)$,
- the latency from $\tau(m_p^1)$ to $\tau(m_p^2)$,
- the latency from $\tau(m_p^2)$ to t_f^k, the publishing time of m_i^k.

The first part can be bounded by the time disparity upper bound of a published set. In [2], a time disparity upper bound of any published set is derived. We use $\overline{\Delta}$ to denote this upper bound, i.e.,

$$\overline{\Delta} = \max_{2 \le n \le N} \left\{ \frac{1}{n} \sum_{n-1 \ \textit{largest}} T_j^U \right\} \tag{1}$$

The second part can be trivially bounded by the maximal timestamp interval of sensor p.

Upper bound on the third part can be obtained by the following lemma.

Lemma 2. *For any selected set S, $\overline{\Delta}$ is an upper bound on its time disparity.*

Proof. [2] shows that $\overline{\Delta}$ is a time disparity bound of a reference set based on its Lemma 3, Lemma 5 and Theorem 1, where the reference set is one of the regular set. By Definition 4 we know the selected set S has the smallest time disparity among all corresponding regular sets. Therefore, $\overline{\Delta}$ must be a bound of a selected set. □

Lemma 3. *For the published set $S_{\textbf{PUB}}$, let $m_{\textbf{P}}$ be the pivot, and t_f be the publishing time. According to the algorithm, t_f must be the arrival time of a message. Let m_l denote this message and l denote the queue. Then $\tau(m_l)$ is bounded by*

$$\tau(m_l) - \tau(m_{\textbf{P}}) \le T_l^U - \max(T_l^L - \overline{\Delta}, 0)$$

Proof. At time t_f, for each non-pivot queue i, let m_i^X be the last message with $\tau(m_i^X) \le \tau(m_{\textbf{P}})$, and m_i^Y be the first message with $\tau(m_i^Y) > \tau(m_{\textbf{P}})$. If m_l is one of m_i^X or it is $m_{\textbf{P}}$, then $\tau(m_l) - \tau(m_{\textbf{P}}) \le 0$ and the lemma holds.

Otherwise m_l is one of m_i^Y and it is not a predicted message. Assume m_l is m_l^Y. If $T_l^L \le \overline{\Delta}$, we need to prove that $\tau(m_l) - \tau(m_{\textbf{P}}) \le T_l^U$. Since the preceding message of m_l

is m_l^X, which is earlier than $m_{\mathbf{P}}$, we have $\tau(m_l) - \tau(m_{\mathbf{P}}) \leq \tau(m_l) - \tau(m_l^X) \leq T_l^U$ and the lemma holds.

The only remaining case is $T_l^L > \overline{\Delta}$, we need to prove that

$$\tau(m_l) - \tau(m_{\mathbf{P}}) \leq \overline{\Delta} + T_l^U - T_l^L \tag{2}$$

Let's consider the situation when the last message before m_l (regardless of which one) arrived. We use t' to denote that time and use primed names to distinguish the corresponding entities at t'. Let S be the selected set at time t_f and S' be the selected set at t'. Note that $m_{\mathbf{P}}{}'$, all $m_i^{X'}$ and $m_i^{Y'}$, are the same as their respective unprimed type, except for $m_l^{Y'}$, which is not m_l but the predicted message corresponding to m_l. We know, at t', the algorithm failed to publish S'. The only possible reason is that S' contains at least one predicted message (line 13). Next we prove by contradiction that, at this time, $m_l^{Y'}$ must be in S'.

Suppose $m_l^{Y'}$ is not in S'. By the definition of predicted message, the time interval between $m_l^{Y'}$ and $m_l^{X'}$ is T_l^L, which can not be greater than interval between m_l^Y and m_l^X. Since $m_l^{X'} = m_l^X$ and $m_{\mathbf{P}}{}' = m_{\mathbf{P}}$, $m_l^{Y'}$ is closer to $m_{\mathbf{P}}{}'$ than m_l^Y to $m_{\mathbf{P}}$. Apart from m_l^Y, all messages are the same as in the primed case. Therefore, the synchronizer is facing the same situation at t' except that $m_l^{Y'}$ is closer to the pivot than m_l^Y. If $m_l^{Y'}$ is not chosen into S', m_l^Y will not be chosen into S either, which causes contradiction to the fact that m_l^Y is actually published. Therefore, $m_l^{Y'}$ must be in S'.

Since S' is a selected set, based on lemma 2, we have

$$\tau(m_l^{Y'}) - \tau(m_{\mathbf{P}}) = \tau(m_l^{Y'}) - \tau(m'_{\mathbf{P}}) \leq \overline{\Delta} \tag{3}$$

Since $m_l^{Y'}$ is a predicted message and m_l^Y is the corresponding real message, we have:

$$\tau(m_l^Y) - \tau(m_l^{Y'}) \leq T_l^U - T_l^L \tag{4}$$

By combining (3) and (4), (2) can be proved. This completes the proof of the lemma. □

Now we can sum the three parts to derive our reaction latency upper bound.

Theorem 1. *For any published message m_i, its reaction latency is upper bounded by*

$$\overline{R} = \overline{\Delta} + \max_{1\leq j\leq N} T_j^U + \max_{1\leq k\leq N}(T_k^U - \max(T_k^L - \overline{\Delta}, 0) + D_k^U) - D_i^L$$

Proof. We use m_i^2 to denote m_i. Let $S^2_{\mathbf{PUB}}$ be its published set, m_p^2 be the pivot, t_f be its publishing time, m_l be the message arrives at t_f. Denote the last published message in the same queue as m_i^1, whose published set is $S^1_{\mathbf{PUB}}$. Let $m_p^1 \in S^1_{\mathbf{PUB}}$ be the message in the same queue as m_p^2. By the definition of pivot, there is no message before m_p^2 when it arrives, which means that it is adjacent to m_p^1. Then we have

$$\begin{aligned}
&R(m_i^2)\\
&= \alpha(m_l) - \alpha(m_i^1)\\
&\leq \tau(m_l) - \tau(m_i^1) + D_l^U - D_i^L\\
&\leq \tau(m_l) - \tau(m_p^2) + \tau(m_p^2) - \tau(m_p^1) + \tau(m_p^1) - \tau(m_i^1) + D_l^U - D_i^L\\
&\leq T_l^U - \max(T_l^L - \overline{\Delta}, 0) + T_p^U + \overline{\Delta} + D_l^U - D_i^L\\
&\leq \overline{\Delta} + \max_{1\leq j\leq N} T_j^U + \max_{1\leq k\leq N}(T_k^U - \max(T_k^L - \overline{\Delta}, 0) + D_k^U) - D_i^L
\end{aligned}$$

□

B. Tightness

In the following, we construct a worst-case system to demonstrate the tightness of our reaction latency upper bound.

The system contains N queues. For $i \neq N-1$, $T_i^U = T$. Let $\delta > 0$ be a small value, $T_{N-1}^L = T - \delta$ and $T_{N-1}^U = T + \delta$. The first message of each queue are sampled sequentially with an interval of $\frac{T}{N}$, i.e., $\tau(m_1^1) = 0, \tau(m_i^1) - \tau(m_{i-1}^1) = \frac{T}{N}$ for $2 \leq i \leq N$. After that, messages in each queue are sampled at their corresponding T_i^U, except for queue $N-1$, whose second message is sampled after interval T and third message is sampled after $T_{N-1}^U = T + \delta$. In this system, the reaction latency of the second message in Q_1 approaches $\overline{R}$ as δ approaches 0. Next we show the calculation detail of this result based on a specific example.

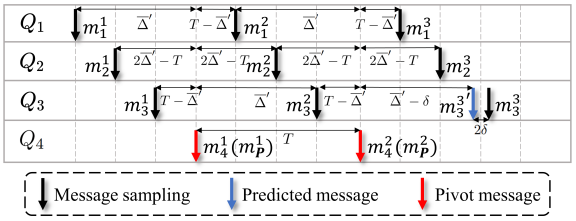

Fig. 4. An example of worst-case reaction latency

The case for $N = 4$ is illustrated in Fig. 4. We use notation $\overline{\Delta}' = \frac{(N-1)T}{N}$ to simplify expression.

We first ignore the influence of delay ($\forall i, D_i^U = 0$). Upon the arrival of $m_4^1(m_{\mathbf{P}}^1)$, the arrived messages form a regular set with a time disparity of $\overline{\Delta}'$. However, the system does not select this set because the interval between the predicted message of Q_3 and $m_{\mathbf{P}}^1$ is $\overline{\Delta}' - \delta$. A regular set comprising the predicted message of each queue has a smaller time disparity. The system will have to wait for m_3^2. Upon the arrival of m_3^2, the system have multiple regular sets with the smallest time disparity $\overline{\Delta}'$. According to Algorithm 1 it selects the earliest set composed of the first messages, specifically including m_1^1. The system encounters a similar scenario upon the arrival of $m_4^2(m_{\mathbf{P}}^2)$. Again, it opts to wait for m_3^3, which may arrive after a smaller time interval (as $m_3^{3'}$). However, upon the arrival of this message, it turns out that the interval between m_3^3 and $m_{\mathbf{P}}^2$ is even larger ($\overline{\Delta}' + \delta$). Consequently, the system selects and publishes the set comprising all second messages m_i^2, specifically including m_1^2. The reaction latency of message m_1^2 is given by:

$$R(m_1^2) = t_f - \alpha(m_1^1) = \tau(m_3^3) - \tau(m_1^1) = 2\overline{\Delta}' + T + \delta$$

Based on (1) we have $\overline{\Delta} = \frac{(N-1)T+\delta}{N}$. Let $\delta \leq \frac{T}{N+1}$ such that $T_{N-1}^L \geq \overline{\Delta}$. The reaction latency upper bound according to Theorem 1 is

$$\begin{aligned}
\overline{R} &= \overline{\Delta} + \max_{1\leq j\leq N} T_j^U + \max_{1\leq k\leq N}(T_k^U - \max(T_k^L - \overline{\Delta}, 0)\\
&= 2\overline{\Delta} + T + 3\delta
\end{aligned}$$

Since the *ApproximateTime* policy utilizes only the timestamps of messages, the delay has no impact on its selecting published sets. Therefore we can arbitrarily assign delay to each message without affecting the above analysis. To achieve the worst case, we set $D_1^L = D_1^U = D_{min}$ and $D_{N-1}^L = D_{N-1}^U = D_{max}$, where D_{min} and D_{max} represent the minimal and maximal latency among all queues. Other queues have arbitrary delays between them. This modification will add the same term $(D_{max} - D_{min})$ to $R(m_1^2)$ and $\overline{R}$. Thus we have

$$\frac{\overline{R}}{R(m_1^2)} = \frac{2\overline{\Delta} + T + 3\delta + D_{max} - D_{min}}{2\overline{\Delta}' + T + \delta + D_{max} - D_{min}} \xrightarrow{\delta \to 0} 1$$

This demonstrates the tightness of our bound.

VI. Experiments

All our experiments are based on the open-source ROS 2 system of version *Iron Irwini*. We conduct experiments on a desktop computer equipped with an Intel(R) Core(TM) i5-11400H CPU running at 2.70GHz. We also use a real autonomous vehicle model to validate the timing defect and exhibit its impact.

There are 4 different experiment settings:

- *Random Delay*: delay time ranges from 0 to 40 ms.
- *Number of Sensors*: sensor number ranges from 3 to 9.
- *Index of Sensor*: different index of sensor evaluated.
- *Period Ratio*: ratio of T^U/T^L ranges from 1.0 to 1.8. Each T_i^L is randomly selected in $[50, 100]$ ms.

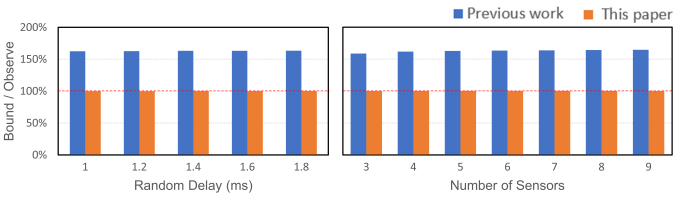

Fig. 5. Evaluation result of worst-case reaction latency

We evaluate the tightness of our reaction latency upper bound using the worst-case scenario described in section V-B.

Fig. 5 shows the results. The height of bar denotes the ratio of the bound to the maximal observed value in all repeated experiments. The upper bound presented in [1] (tagged as *Previous work*) exhibits significant overestimation. In contrast, the maximum overestimation observed for our result (tagged as *This paper*) is 0.3%, demonstrating its tightness. This small overestimation may stem from the value of $\delta = T/100$. In addition, among all experiments, our bound is higher than observed, which verifies its safety.

We conduct further experiments under randomly configured systems to evaluate the accuracy and robustness of our bound. The results are shown in Fig. 6. Our bound exhibits low pessimism across all settings, with an average overestimation of 20%. The upper bound in [1] has an relatively high average overestimation of 100%. The remaining overestimation of our bound can be attributed to the fact that randomly configured system can hardly yield worst-case scenarios.

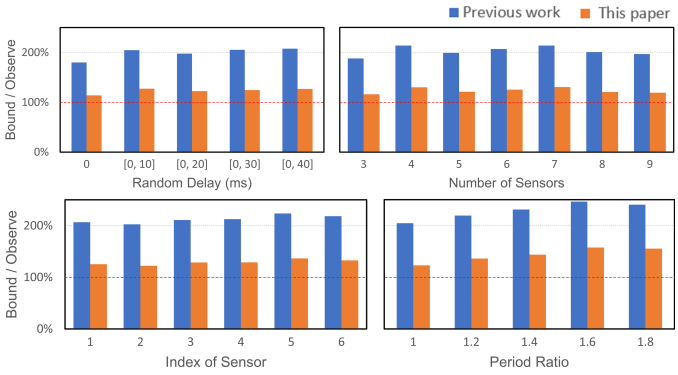

Fig. 6. Evaluation result of general-case reaction latency

VII. Conclusion

In this paper we present a safe and tight reaction latency upper bound of the *ApproximateTime* synchronization policy in ROS. A specific worst-case system is constructed to prove its tightness. Experiments on real ROS system are conducted to evaluate our bound. The results verifies the safety, tightness and robustness of the proposed reaction latency upper bound.

Acknowledgment

The first and third authors are partly funded by the National Key R&D Program of China under grants No. 2022YFA1005101 and the NSFC under grant No. 62192732. The second and fourth authors are partially supported by Hong Kong GRF under grant no. 15206221 and 11208522.

References

[1] R. Li, X. Jiang, Z. Dong, J.-M. Wu, C. J. Xue, and N. Guan, "Worst-case latency analysis of message synchronization in ROS," *RTSS*, 2023.
[2] R. Li, N. Guan, X. Jiang, Z. Guo, Z. Dong, and M. Lv, "Worst-case time disparity analysis of message synchronization in ROS," in *RTSS*, 2022.
[3] R. Li, Z. Dong, J.-M. Wu, C. J. Xue, and N. Guan, "Modeling and property analysis of the message synchronization policy in ROS," in *MOST*, 2023.
[4] "ROS." [Online]. Available: https://www.ros.org/
[5] "ROS2 Iron." [Online]. Available: https://docs.ros.org/en/rolling/Releases/Release-Iron-Irwini.html#
[6] "The *ExactTime* Policy." [Online]. Available: http://wiki.ros.org/message_filters#ExactTime_Policy
[7] "The *ApproximateEpsilonTime* Policy." [Online]. Available: https://github.com/ros2/message_filters/blob/rolling/include/message_filters/sync_policies/approximate_epsilon_time.h
[8] "The *LatestTime* Policy." [Online]. Available: https://github.com/ros2/message_filters/blob/rolling/include/message_filters/sync_policies/latest_time.h
[9] "The *ApproximateTime* Policy." [Online]. Available: http://wiki.ros.org/message_filters#ApproximateTime_Policy
[10] J. Sun, T. Wang, Y. Li, N. Guan, Z. Guo, and G. Tan, "Seam: An optimal message synchronizer in ROS with well-bounded time disparity," in *RTSS*, 2023.
[11] J. Peršić, L. Petrović, I. Marković, and I. Petrović, "Online multi-sensor calibration based on moving object tracking," in *Adv. Robotics*, 2021.
[12] M. R. Nowicki, "Spatio-temporal calibration of camera and 3d laser scanner," in *IEEE Robot. Autom. Lett.*, 2020.
[13] X. Jiang, D. Ji, N. Guan, R. Li, Y. Tang, and Y. Wang, "Real-time scheduling and analysis of processing chains on multi-threaded executor in ROS 2," in *RTSS*, 2022.
[14] H. Teper, M. Günzel, N. Ueter, G. von der Brüggen, and J.-J. Chen, "End-To-End Timing Analysis in ROS2," in *RTSS*, 2022.