Poster Abstract: Extending Schedule-Abstraction Graph for Event-Triggered Response-Time Analysis

Ruide Cao
SUSTech
Heyuan DET
Shenzhen, China

Qinyang He
Nankai University
Tianjin, China

Yi Wang
SUSTech
Heyuan DET
Peng Cheng Laboratory
Shenzhen, China

Zhuyun Qi
Tsinghua SIGS, Tsinghua University
Peng Cheng Laboratory
Shenzhen, China

ABSTRACT

For cyber-physical systems, the predictability of their physical behaviors needs to be ensured by the determinism of cyberspace. Response-time analysis (RTA) can theoretically provide this determinism by analyzing the temporal properties of demands. However, the state-space explosion problem makes it challenging to do exact and sustainable RTA for non-preemptive systems where both release jitter and execution time variation exist, particularly when the system has event-triggered (ET) jobs. To address this issue, we propose an ET-enabled RTA based on the schedule-abstraction graph and preliminarily verify its effectiveness and scalability.

KEYWORDS

Event-triggered, response-time analysis, schedule-abstraction graph

1 INTRODUCTION

IT/OT convergence is evolving toward increasingly diverse and complex scenarios under the Industry 4.0 trend, where the behaviors of cyber-physical systems are always expected to be predictable and robust, especially in industrial environments. For instance, industrial IoT gateways represent standard hard real-time systems, requiring assurances of worst-case latency performance. With the ability to theoretically guarantee time-correctness, response-time analysis (RTA) has been widely applied to these systems.

Schedule-abstraction graph (SAG) [4] is an exact and sustainable RTA framework for non-preemptive situations in processor and network contexts. It is proved to be at least 3000 times faster than other exact UPPAAL-based RTAs [8]. As a promising RTA approach, it has been extended and applied to analyze the CAN message schedules with retransmissions [5] and to make the worst-case latency analysis of TSN with time-aware shapers [7]. Recently, an extended SAG that supports dynamic job sets has been proposed in [2], which allows the emerging jobs to be considered during the SAG construction. However, the case where jobs were abandoned for execution was not discussed so far.

While the time-triggered (TT) mechanism has a stable temporal behavior by periodically releasing jobs within predetermined time ranges, it tends to lead to excessive consumption of computational and communication resources due to its lack of flexibility [1]. With flexibility that helps to enhance the system performance, event-triggered (ET) jobs are not necessarily triggered in each hyperperiod but only when certain conditional events are met. Therefore, non-abandoning SAG does not apply to systems containing ET jobs.

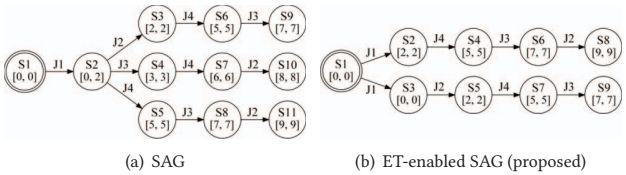
This poster proposes an ET-enabled SAG that can take into account the case where jobs are abandoned by adding a new vertex for each ET job during the construction of the SAG. ET-enabled SAG can thus be applied to do RTA for ET jobs containing systems. Two examples were attached to illustrate the difference between the original SAG and the ET-enabled SAG and show the effectiveness of ET-enabled SAG. Finally, experiments were conducted on task sets constructed using real-world application characteristics, and the scalability of the proposed ET-enabled SAG was proved.

2 SYSTEM MODEL

Consider each job J_i in the job set $\mathcal{J}$ can arrive at $r_i \in [r_i^{min}, r_i^{max}]$ and be finished at the time cost of $C_i \in [C_i^{min}, C_i^{max}]$ with its absolute deadline and fixed priority denoted by d_i and p_i. SAG represents all possible execution scenarios by a directed acyclic graph where each system state is a vertex v_k labeled with the earliest time e_k and the latest time l_k to reach it, and each scheduling decision is an edge labeled with the job selected for execution.

The original SAG has two phases to construct: the expansion phase and the merging phase. When taking ET jobs into account, an intuitive operation is to set the C_e^{min} of every ET job J_e directly to 0. However, this operation may make SAG not *exact*. Although it is still exact according to Def. 3 and Def. 4 in [6], these definitions are no longer valid since the possible execution time is no longer contiguous. Whenever the original C_e^{min} is greater than 1, setting it to 0 introduces the $[1, C_e^{min} - 1]$ segment, which will not happen, into the subsequent graph expansion. For example, S7 in Fig. 1(a) caused a missed deadline on J_4 and made the job set unschedulable.

To address this issue, we additionally use E_i to denote whether J_i is an event-triggered job. Thus J_i can be described as $([r_i^{min}, r_i^{max}], [C_i^{min}, C_i^{max}], d_i, p_i, E_i)$. Instead of setting C_e^{min} to 0, we leave C_e^{min} unchanged at first and create an additional vertex for every $E_e = 1$ eligible successor J_e with both C_e^{min} and C_e^{max} set to 0 during the



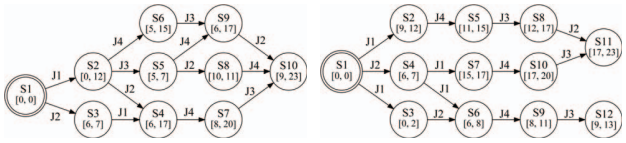
(a) SAG (b) ET-enabled SAG (proposed)

Figure 1: $\{J_1 = ([0,0],[2,2],5,1,1),\ J_2 = ([0,0],[2,2],10,4,0),\ J_3 = ([1,1],[2,2],10,3,0),\ J_4 = ([2,2],[3,3],5,2,0)\}$ **as Job set.**

Corresponding author: Zhuyun Qi (qizy@pcl.ac.cn). This work is supported by the National Key Research and Development Program of China (2020YFB1806400), the Major Key Project of Peng Cheng Laboratory (PCL2023A03), and the Guangdong High-Level Talents Special Support Program (2021TX05X205).

979-8-3503-6201-5/24/$31.00 ©2024 IEEE
DOI 10.1109/IPSN61024.2024.00055

expansion phase of SAG construction. Since both vertices produced by the same J_e may be merged into the same existing vertex in the merging phase, we also add an edge merging operation for the case. Fig. 1(b) correctly describes all possible execution scenarios with no missed deadline, and the job set is deemed schedulable.



(a) SAG (b) ET-enabled SAG (proposed)

Figure 2: $\{J_1 = ([0, 2], [9, 10], 20, 1, 1)$, $J_2 = ([1, 2], [5, 6], 30, 4, 0)$, $J_3 = ([4, 5], [1, 2], 30, 3, 0)$, $J_4 = ([3, 6], [2, 3], 30, 2, 0)\}$ **as Job set.**

A more complex example is shown in Fig. 2, where release jitter and execution time variation exist in all jobs. The fixed priority scheduling policy is applied. Both SAG and ET-enabled SAG start from the initial root state S1 labeled $[0, 0]$, which implies that the system is idle and no job is executed. They chose J_1 and J_2 as the first jobs since J_1, J_2 were released first. As $E_1 = 1$, J_1 is an ET job and may not be triggered in some hyperperiods. While SAG only creates a single vertex for the J_1 executed state with $C_1^{min} = 0$, ET-enabled SAG creates two vertices corresponding to cases where J_1 is executed and abandoned. As a result, S2 in Fig. 2(a) has an extra outgoing edge labeled J_3 compared to S2 and S3 in Fig. 2(b). An unreachable state, S5, has been introduced to SAG, and the best-case response time of J3 is prematurely analyzed as $e_5 - r_3^{min} = 1$.

3 PRELIMINARY RESULTS

Since the scenarios where an ET job is not triggered are represented as new vertices in our proposed ET-enabled SAG, the state explosion problem could be exponentially serious and thus must be considered. We conducted experiments on an i7-12700KF CPU to preliminarily evaluate its scalability compared to the original SAG.

According to the characteristics of a real-world automotive engine control application from Bosch provided by Kramer et al. [3], the typical total number of jobs is between 1000 and 1500, with TT jobs accounting for 85% and ET jobs accounting for 15%. So we randomly generated 1000 jobs for each job set, with $r_i^{min} \in [1, 9900]$, $r_i^{max} \in [r_i^{min} + 0, r_i^{min} + 9]$, $C_i^{min} \in [2, C_U]$, $C_i^{max} \in [C_i^{min} + 1, C_i^{min} + 4]$, $d_i = 9999$, $p_i \in [1, 10]$ for each job J_i. Moreover, we reach the desired utilization by controlling the value of C_U.

Fixing the ratio of ET jobs at 15%, Fig. 3(a) shows how big the SAGs and the ET-enabled SAGs are by their vertex number under different utilization cases, and the corresponding construction time is shown in Fig. 3(b). When SAG becomes progressively larger with the rise of utilization, the ET-enabled SAGs are stabilized at about 5% larger (4.61% on average) and take about 11.0% (10.96% on average) more time to construct. These ratios do not show a clear trend across different utilization cases but rather fluctuate slightly around the average, which means that the proposed scheme is unaffected or minimally affected by utilization changes.

Then, we fix the utilization of job sets at 60%. Fig. 3(c) shows the size of the SAGs and the ET-enabled SAGs with different ET ratio settings, and the corresponding construction time is shown in Fig.

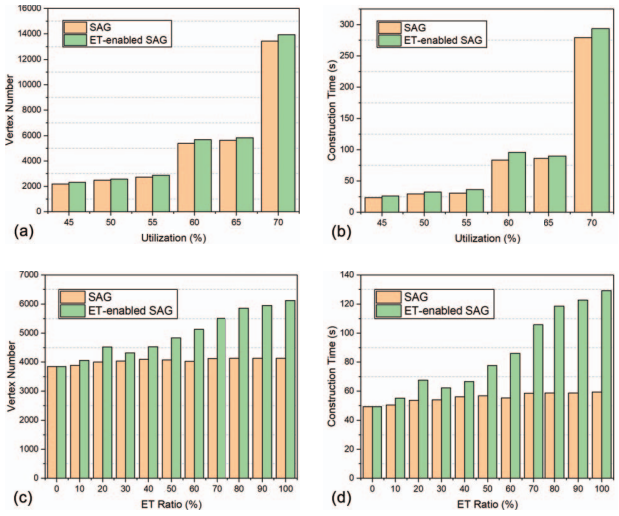


Figure 3: Experimental results.

3(d). When the ET ratio is 0%, SAG and ET-enabled SAG are the same size and took almost the same time to construct. When the ET ratio reaches 100%, ET-enabled SAG has 48.18% more vertices than SAG, and its construction time is up to 2.17 times that of SAG. These ratios increase near-linearly as the ET ratio rises, and we consider this growth level acceptable.

4 FUTURE WORK

With the effectiveness and scalability of ET-enabled SAG preliminarily verified by two specific examples and experimental results, we plan to take a further step toward SAG-based RTA for non-zero, non-contiguous job execution time scenarios. Furthermore, in addition to empirical evaluation, it would be desirable to demonstrate a theoretical upper bound on its required computational overhead.

REFERENCES

[1] Xiaohua Ge, Qing-Long Han, Xian-Ming Zhang, and Derui Ding. 2021. Dynamic event-triggered control and estimation: A survey. *International Journal of Automation and Computing* 18, 6 (2021), 857–886.

[2] Pourya Gohari, Jeroen Voeten, and Mitra Nasri. 2023. Response-time Analysis of Fault-Tolerant Hard Real-Time Systems Under Global Scheduling. In *2023 IEEE 29th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 263–264.

[3] Simon Kramer, Dirk Ziegenbein, and Arne Hamann. 2015. Real world automotive benchmarks for free. In *6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, Vol. 130.

[4] Mitra Nasri and Bjorn B Brandenburg. 2017. An exact and sustainable analysis of non-preemptive scheduling. In *2017 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 12–23.

[5] Mitra Nasri, Arpan Gujarati, and Björn B Brandenburg. 2018. Using Schedule-Abstraction Graphs for the Analysis of CAN Message Response Times. In *Proceedings of International Workshop on Security and Dependability of Critical Embedded Real-Time Systems*.

[6] Sayra Ranjha, Geoffrey Nelissen, and Mitra Nasri. 2022. Partial-Order Reduction for Schedule-Abstraction-based Response-Time Analyses of Non-Preemptive Tasks. In *2022 IEEE 28th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 121–132.

[7] Srinidhi Srinivasan, Geoffrey Nelissen, and Reinder J Bril. 2021. Work-in-progress: Analysis of tsn time-aware shapers using schedule abstraction graphs. In *2021 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 508–511.

[8] Beyazit Yalcinkaya, Mitra Nasri, and Björn B Brandenburg. 2019. An exact schedulability test for non-preemptive self-suspending real-time tasks. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1228–1233.