공학석사학위논문

Functionally and Temporally Correct Simulation for ROS2 Cyber Systems of Automotive Systems

ROS2기반의 자동차 사이버 시스템을 위한 기능적/시간적 정확성을 보장하는 실시간 시뮬레이션 기법

2021년 2월

서울대학교 대학원
컴퓨터공학부
박 성 현

Functionally and Temporally Correct Simulation for ROS2 Cyber Systems of Automotive Systems

ROS2기반의 자동차 사이버 시스템을 위한 기능적/시간적 정확성을 보장하는 실시간 시뮬레이션 기법

지도교수 이 창 건

이 논문을 공학석사 학위논문으로 제출함

2020년 11월

서울대학교 대학원

컴퓨터공학부

박 성 현

박성현의 공학석사 학위논문을 인준함

2020년 12월

위 원 장 하 순 회 (인)

부위원장 이 창 건 (인)

위 원 김 태 현 (인)

Abstract

Functionally and Temporally Correct Simulation for ROS2 Cyber Systems of Automotive Systems

Seonghyeon Park
Department of Computer Science and Engineering
The Graduate School
Seoul National University

This dissertation proposes an approach for functionally and temporally correct simulation of cyber system based on ROS2 framework. In the previous work, the simulation approach was proposed that overcomes the limitations, which only guaranteeing the functional correctness of the existing simulation approach by guaranteeing the temporal correctness and simultaneously performs the task efficiently by reordering jobs. Recognizing that the ROS2 cyber system differs from the traditional automotive cyber systems, this dissertation can be applied to the ROS2 cyber system while maintaining the key idea of the previous simulation approach. In the proposed approach, a system model for ROS2 cyber system is defined. Based on this, the cyber system's schedule is predicted, and a precedence relationship graph is generated so that the existing simulation technique can be applied. The proposed method measures the simulation capacity, together with other simulation algorithms, through a

randomly generated workload, and it is shown that the proposed approach has the highest simulation capacity in a single core simulator. Therefore, the existing functional/temporally correct simulation approach can be applied to the cyber system of automotive system based on ROS2 framework, and by utilizing this, it is possible to correctly and effectively simulate the ROS2 cyber system.

keywords : Automotive System, Real-Time Simulation, ROS2 Framework

Student Number : 2019-23556

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

For automotive system developers, it is important to validate the system performance at the design phase[3]. In the automotive industry, simulation approach is commonly used for the validation of the automotive systems such as Simulink[4]. However, if there is a gap between real performance and simulation result, they need to go through the process of redesigning and implementing the system which increases the cost for developing the system. For example, Figure 4 shows the difference between the simulated results and real performance by Simulink[4], which cannot guarantee temporal correctness. Therefore, we need a simulation approach that guarantees not only functional correctness but also temporal correctness.

For this, K.S. We et al. [5] already addressed this problem with a novel simulation approach which can guarantee both of functional correctness and temporal correctness. By maintaining the same data and time as the real cyber system only at the physical interaction point with the physical system, the approach can guarantee not only correct simulation but also efficient simulation by reordering the jobs to schedule more efficiently.

Since, the ROS(Robot Operating System)[6] and ROS2(the new released version supporting real-time features)[7] have become popular in the automotive industry because of its various libraries to develop robotic system and enormous community to share information for system developers, we intend to use this simulation approach for the automotive systems designed by ROS2 framework which is supporting the real-time features.

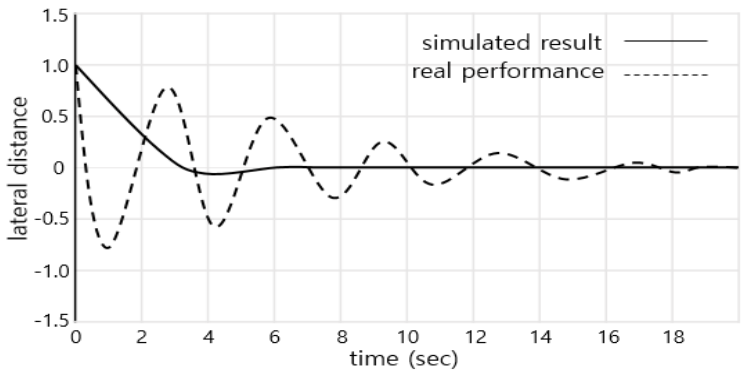


Figure 1: Gap between predicted performance and real performance of LKAS [1]

However, we cannot apply the simulation approach directly to the cyber system based on ROS2 because of different execution behaviors of functions. In the ROS2 system, we need to consider one more layer that affect to the execution behaviors. Therefore, we propose extended simulation approach for ROS2 cyber-system by considering the difference of execution behavior. Our contribution is to keep the functional and temporal correct simulation approach to the ROS2 cyber system by analyzing the execution behavior of ROS2 cyber system and defining new features for adapting existing simulation algorithm.

Our Contributions:

- We propose a system model for the cyber systems based on the ROS2 framework, and to simulate this, we analyze the execution behavior of the ROS2 cyber system.
- We show the highest simulation capacity in the simulator PC which has an uniprocessor by the simulation results of synthetic workloads so that we keep the existing simulation approach on the ROS2 cyber system.

1.2 Organization

This paper is organized as follows. In Section 2, we review the functionally and temporally correct simulation approach and define new system model for cyber systems based on ROS2 framework. Then, Section 3 explains our proposed approach. In Section 4, we show our experiment results. Finally, Section 5 concludes the paper.

2 Backgrounds

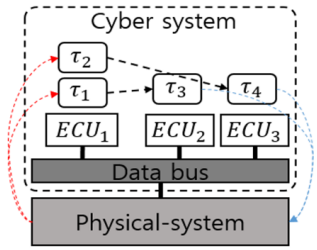
In this section, we review functionally and temporally correct simulations[5] and ROS2 scheduling[2].

2.1 Overview of Functionally and Temporally Correct Simulation

In the previous simulation approach, the cyber-systems consist of multiple ECUs which have a specific preemptive fixed priority (i.e., RM[8]) scheduler. Those scheduler schedules periodic task τ_i, represented as five properties:

$$\tau_i = (F_i, \Phi_i, P_i, C_i^{bcet}, C_i^{wcet})$$

where F_i denotes the function of control algorithm which is executed by τ_i, Φ_i is the offset from the system start time, P_i is the period of τ_i. C_i^{bcet} denotes the best execution time of τ_i, C_i^{wcet} denotes the worst case execution time of τ_i. The cyber system of an automotive system can be given as Figure 2 shows. In the Figure 2(a), data read interactions from the physical system are denoted by red dotted arrows and data write interactions to the physical system are denoted by blue dotted arrows. For the data producer consumer relations, we denoted the relation by black dotted arrows. For the cyber system, tasks parameters are given as Figure 2(b). From the parameters, we generate an execution scenario of the cyber system as shown in Figure 3(a). The Figure 3(a) shows the schedule of the real cyber system for a hyper period of tasks which is 8. The upper arrows in physical system denote the incoming data and red arrows denote the data interaction point for reading and blue arrows denote data interaction point for writing. J_{ij} in the box denotes the j-th instance of τ_i. From the schedule, we

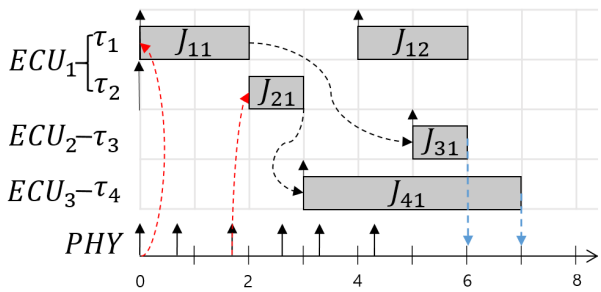


(a) Cyber system design

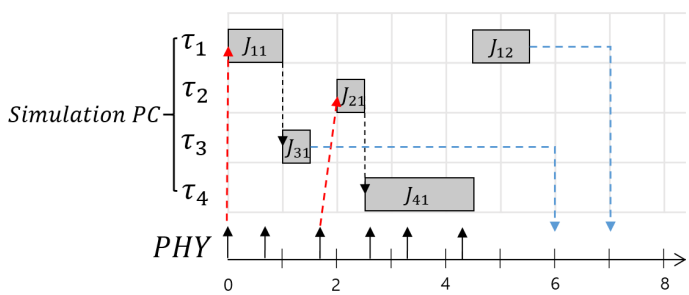
		Φ_i	P_i	C_i^b	C_i^w
ECU_1	τ_1	0	4	1	3
	τ_2	0	8	1	2
ECU_2	τ_3	4	8	1	2
ECU_3	τ_4	3	8	4	5

(b) Task parameters

Figure 2: Example cyber system of automotive system



(a) Execution scenario of real cyber system



(b) Execution scenario of simulation

Figure 3: Execution scenario and simulation scenario of example automotive system

can know each job's release time and start time range and finish time range. Using the schedule, we construct a graph for representing precedence relation between the jobs. As the above simulation scenario, we assume the following assumptions for the

simulation:

- **Execution time mapping function:** For the same function F_i of task τ_i, PC can execute the function more faster than ECU. For the execution times, there is a relation between real cyber system's execution time and simulation execution time. We call this mapping function and represent as $e_{ij}^{real} = M_i(e_{ij}^{sim})$ where M_i represents the execution time mapping function. e_{ij}^{real} represents the execution time on the ECU and e_{ij}^{sim} represents the execution time on the simulation PC.

- **Most Recent Data Use:** Each data is updated in a single memory buffer. Therefore, the memory buffer is overwritten by the most recent data which is produced by a job or physical systems.

- **Entry Read and Exit Write:** Each job read all the data at the entry of the execution and write all the output at the exit of the execution.

- **Tagged Data Read:** Real cyber system keep only the most recent data, but the simulation PC can log all the data history with tag which has timestamp and producer information. Therefore, simulation PC can execute a job after data logged.

- **Delay Data Write:** Real cyber system has to write output when execution is done. But the simulation PC can delay write time of job's output.

In the previous work, the simulation PC runs the jobs in the graph with effective EDF scheduling policy and Simulation PC has a single core. From the above assumptions, it can delay data writing or read data with the same data with its real instance in the real cyber system. For this, simulated job with red arrow must start

later than its real start time and the job with blue arrow must finish faster than its real finish time. As a result, we can see that red arrow and blue arrow points same time as shown. This means that we keep the same data and time with real cyber system with enjoying the freedom of job scheduling. For keeping the functional and temporal correctness only at physical interaction points while enjoying the freedom of executing job scheduling, we have to consider below constraints:

- **Physical-read constraint:** If a job J_{ij} which reads data from the physical system, the simulation PC must run the job later than its actual start time on the real cyber system, i.e.,

$$t_{ij}^{S,sim} \geq t_{ij}^{S,real} \tag{1}$$

  where $t_{ij}^{S,sim}$ and $t_{ij}^{S,real}$ represent the simulated start time of J_{ij} on the simulation PC and the actual start time on the real cyber system.

- **Physical-write constraint:** If a job J_{ij} which writes data to the physical system, the simulation PC must finish the job before its actual finish time on the real cyber system, i.e.,

$$t_{ij}^{F,sim} \leq t_{ij}^{F,real} \tag{2}$$

  where $t_{ij}^{F,sim}$ and $t_{ij}^{F,real}$ represent the simulated finish time of J_{ij} on the simulation PC and the actual finish time on the real cyber system.

- **Producer/consumer constraint:** If a pair of jobs, $J_{i'j'}$ and J_{ij}, which $J_{i'j'}$ is a producer job of J_{ij} on the real cyber system, the simulation PC must finish $J_{i'j'}$

before the actual start time of J_{ij} on the real cyber system, i.e.,

$$t_{i'j'}^{F,sim} \leq t_{ij}^{S,real} \tag{3}$$

2.2 ROS2 Scheduling

In the ROS2 cyber system, despite assuming to use the ECUs using the same scheduler, it is essential to consider that the function execution behavior is different than traditional cyber systems of automotive systems. In the traditional cyber systems, tasks was scheduled by OS scheduler so that we can regard task as process. However, in the ROS2 process, which is running on the OS scheduler, there is a scheduler in the main thread called 'Executor' which is the charge of executing the functions called callback[2]. There are two types of implementation of executor in the ROS2: single-threaded, multi-threaded. Single threaded executor executes callbacks one by one, in other words, it is non-preemptive scheduler. Executor has a set of callbacks to run. For executing the functions on the executor, there are four types of callbacks: Timer, Subscriber, Service, Client. Timer callback is periodically released by system timer, but subscriber call-back is released by its publisher callback which writes data to its subscriber callbacks. As shown in Figure 4, the executor is updated whenever there are no callbacks to run in the executor by looking for ready callback in communication layer. However, timer callbacks are not managed by communication layer. When the ready set is updated, any timer callback whose timer is expired is executed. If there is no timer callback, executor searches subscriber callbacks in the set and execute them all. If there is no subscriber callback, executor searches service call-

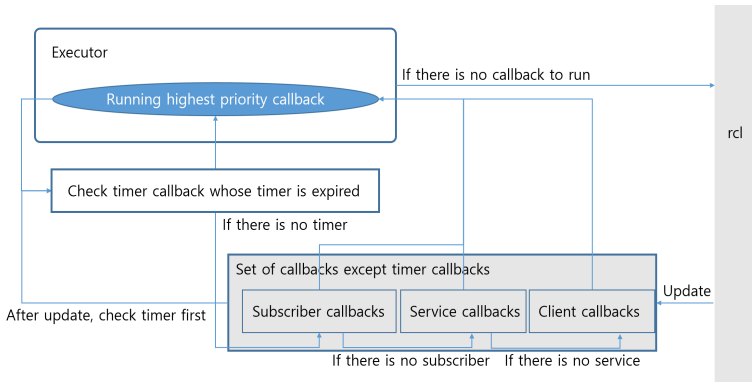


Figure 4: Executor's execution behavior [2]

backs in the set and execute them all. For the last, client callbacks are also executed by same way. Note that timer callback is always checked between the time of running callbacks. Therefore, there is a scheduling policy in the executor for executing those callback types. Timer callback has highest priority, but we have to know how we distinct the priority if callbacks to run are same callback type. For this, they have registration order which means that when you add the callbacks to executor, registration time of the callbacks will be priority of the callbacks. Then we can conclude there are two policies: callback types, registration order.

- **Callback Type:** ROS2 has four callback type: timer, subscriber, service, client. Executor considers callback type first. And their priority is given in the order listed above.

- **Registration Order:** In the ROS2, the nodes which is callback container are registered by executor. Therefore, even for callbacks of the same type, the priority depends on the order in which they were registered first.

3 Proposed Approach

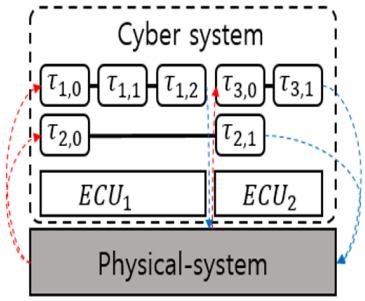
In this section, we first define the system model for ROS2 cyber system and adapt the functionally and temporally correct simulation approach to the ROS2 cyber system.

3.1 System Model for ROS2 Cyber System

In this paper, we assume that each ECU runs only one ROS2 process and we use single threaded executor. By assuming this, we can neglect OS scheduler and focus on non-preemptive execution behavior. We use only timer, subscriber call-back for the system model and assume that all the callback has fixed execution time. Therefore, we can consider these precedence relations as a chain of callbacks called "transaction". Each transaction denoted by T_i consists of a timer callback denoted by $\tau_{i,0}$ at the beginning and subscriber callbacks denoted by $\tau_{i,j}$ at the rest as shown in Eq. 4.

$$T_i = \{\tau_{i,0}, \tau_{i,1}, \tau_{i,2}, \ldots, \tau_{i,j}\} \tag{4}$$

Furthermore, we should consider the executor's scheduling policy. In the executor, there is a ready queue for released jobs called "ready-set" [2]. A job instance of every callback becomes ready state when their message arrived, or their timer expired. For those ready jobs, there are data structures for each of them to storing their jobs. By inspecting those data structures, executor decide which job to execute first with two scheduling policy: Hierarchical callback scheduling, Registration order priority assignment. By hierarchical callback scheduling, a timer call-back always has a higher priority than a subscriber call-back. If the call-back type is same, priorities are assigned according to the registration order by registration order priority assignment



(a) ROS2 cyber system design

Timer Callback

	Φ_i	P_i	C_i	RO_i
ECU_1 $\tau_{1,0}$	0	10	2	1
ECU_1 $\tau_{2,0}$	0	5	2	2
ECU_2 $\tau_{3,0}$	0	10	3	1

Subscriber Callback

	C_i	RO_i
ECU_1 $\tau_{1,1}$	2	1
ECU_1 $\tau_{1,2}$	2	2
ECU_2 $\tau_{2,1}$	3	1
ECU_2 $\tau_{3,1}$	2	2

(b) Task Parameters

Figure 5: Example ROS2 cyber system of automotive system

[2]. For the data exchanges between callbacks and physical interactions, we assume that there are no transmission delay so that we do not consider the data transmission delay in this paper.

3.2 Offline Phase

To represent the simulation steps, we design an example of ROS2 cyber system which has two ECUs: $\{ECU_1, ECU_2\}$ and three transactions $T_1 = \{\tau_{1,0}, \tau_{1,1}, \tau_{1,2}\}$, $T_2 = \{\tau_{2,0}, \tau_{2,1}\}$ and $T_3 = \{\tau_{3,0}, \tau_{3,1}\}$. For the example cyber system, Figure 5(a) is given. In the Figure 5(b), for the timer callbacks, Φ_i is the task offset, P_i is the period of $\tau_{i,0}$, C_i is the constant execution time of $\tau_{i,0}$ and RO_i is the registration order. For the subscriber callbacks, we only use C_i and RO_i.

We can get the hyper period with the timer callbacks periods and in the example case, it is 10. For this, we generate a schedule of real cyber system for 10-time units as shown in Figure 6 before starting the simulation (offline phase). In the schedule, we represent jobs denoted by $J_{i,j,k}$ which means k-th instance of task $\tau_{i,j}$ in the transaction T_i. As shown in Figure 6, we denote only timer callbacks release time

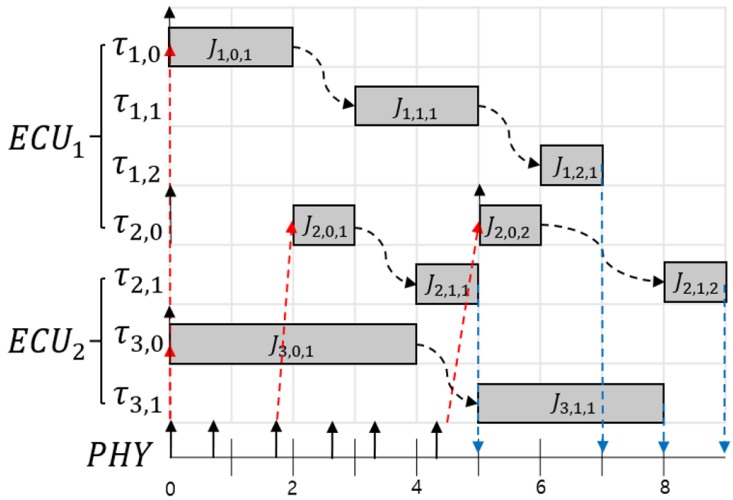


Figure 6: Execution scenario of ROS2 cyber system

as upper arrows. For detail executor's behavior, we show snapshots of executor as shown in Figure 7. We abstract executor's callback queue as a box container, when the job instance of the callback released, then a box for representing the instance fill the mapped container. As we assume that we use single-threaded executor, the current running job cannot be interrupted. At the time unit 0, there are two jobs $J_{1,0,1}$ and $J_{2,0,1}$ released, the executor runs higher priority job $J_{1,0,1}$ as shown in Figure 7(a). Then, $J_{1,0,1}$ creates $J_{1,1,1}$, and $J_{2,0,1}$ starts. At this time, $J_{1,1,1}$ cannot enter readySet because $J_{2,0,1}$ is running as shown in Figure 7(c). At time unit 3, when $J_{2,0,1}$ finished, $J_{2,1,1}$ is generated, but generated in ECU_2. At this time, ECU_1 is in the idle state, and the executor moves $J_{1,1,1}$ to readySet, and because there is only one job in readySet, it is executed immediately. When $J_{1,1,1}$ finishes the execution, $J_{1,2,1}$ is created, but $J_{2,0,2}$ is released at the same time, and $J_{2,0,2}$ is executed immediately because timer callbacks always take precedence over subscriber callbacks. On the other hand, in ECU_2, since $J_{3,0,1}$ is executed up to time unit 4, $J_{2,1,1}$ cannot enter readySet and must

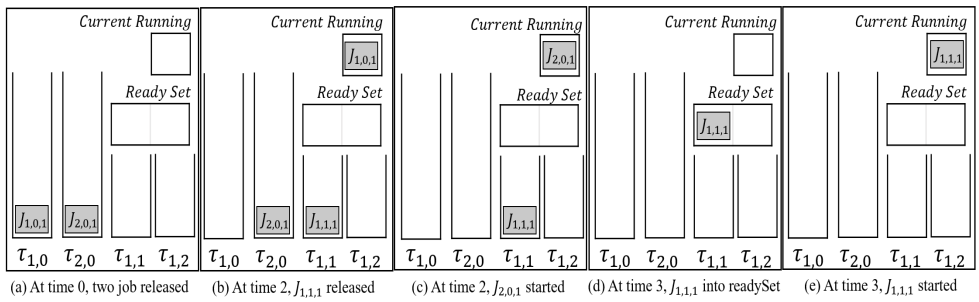


Figure 7: Snapshots of Executor behavior

be pending. When $J_{3,0,1}$ is finished, $J_{3,1,1}$ is created, and ECU_2 becomes idle, so $J_{3,1,1}$ and $J_{2,1,1}$ enter readySet at the same time. Comparing the RO_i of $J_{2,1,1}$ and $J_{3,1,1}$, $J_{2,1,1}$ is higher, so $J_{2,1,1}$ is executed first, followed by $J_{3,1,1}$. Since we assume that a fixed execution time for the tasks, the schedule for one hyper period will be repeated infinitely. Therefore, we can get all the jobs $J_{i,j,k}$'s release time, start time, finish time by generating the schedule.

By the schedule in Figure 6, we have all jobs' release time, start time and finish time. for the functionally and temporally correct simulation, we need to know actual start time and finish time for the physical read constraint and physical write constraint respectively. However we already have these values so we don't have to consider the jobs which can affect the actual start time and finish time. For the producer consumer constraint, the jobs in the same transaction can only be producer. Thus, the precedence graph only consider the producer consumer relation of the transaction. Regarding this, we construct the job precedence graph G=V, E called "Offline guider" as DAG(Directed Acyclic Graph). V is the set of nodes which represent the jobs and E is the set of directed edges which represents precedence relation between jobs. In the offline guider, "R" mark in the upper left side of the nodes denote read constrained

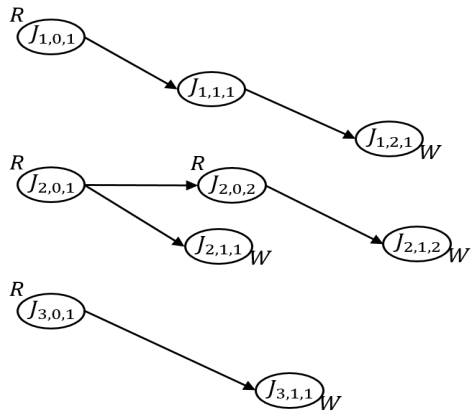


Figure 8: Construction of offline guider

jobs, and "W" mark in the lower right of side of the nodes denote write constrained jobs. As above mentioned, we already know the actual start time and finish time so that there are no the other precedence edges for representing affection for start time and finish time. Thus, in the ROS2 system with fixed execution time tasks, there are no precedence edges are connected only for same transaction or its next period job instances as shown in Figure 8.

3.3 Online Phase

To simulate the jobs in the offline guider, first we assign new deadlines for the jobs in offline guider to keep the write constraint which simulated job should be finished before the actual finish time. For assigning effective deadline we set the effective deadline as their actual finish time for write constrained jobs, while setting other jobs' deadline as infinite. Then, we back-trace write constrained jobs predecessors

and set their deadline as effective deadline as shown in Eq. 5, 6.

$$t_{i,j,k}^{D,sim} = \begin{cases} t_{i,j,k}^{F,real} & \text{for write constrained job} \\ \infty, & \text{otherwise} \end{cases} \tag{5}$$

$$t_{i,j,k}^{D,sim} = \min(t_{i,j,k}^{D,sim}, \min_{\forall J_{i,l,k}^{S} \in T_i}(t_{i,l,k}^{D,sim})) \tag{6}$$

Then, we push the jobs without any precedence edge to the simulation ready queue. However, if the job has read constraint, we first check whether the job satisfy its read constraint. If the simulation time has not yet reached the real start time of the job, pushing to the ready queue is suspended. For executing the ready jobs, we use three simulation approaches: AllSync, Ours, TrueTime [5]. Considering these simulation approaches, we generate a job of next hyper period whenever a job is finished on simulation PC.

- **Ours:** Ours is to execute the jobs in the ready queue by effective EDF scheduling policy. Note that the optimal job scheduling algorithm for the jobs with precedence constraints and effective deadline on uniprocessor is the preemptive EDF scheduling [5, 9].

- **AllSync:** The AllSync approach is the easiest simulation approach and follows the execution order of the real cyber system as much as possible while keeping its real start time for all jobs. In this case, it cannot have job scheduling freedom and hard to simulate enormous task set.

- **TrueTime:** In this paper, we use an extended version of TrueTime approach[5]. In the approach, only jobs with physical interaction points are executed later

than or equal to the real start time. Thus, it has a small freedom to ordering jobs with no physical interaction points. Note that if all the task in the task set are timer callback, then TrueTime approach become same with AllSync approach.

We show our approach's and AllSync execution scenario with the above example case of ROS2 cyber system by showing the two schedules of a hyper period: AllSync approach, our approach as shown in Figure 9(a) and Figure 9(b). In the Figure 9(a), we represent the deadline miss of $J_{1,2,1}$ which is 7 at the Figure 9(b) as its real finish time. Thus, we can say that this task set cannot have a feasible schedule with the All-Sync approach. On the other hand, we swap the scheduling approach to ours which is based on effective EDF scheduling policy and we represent there are no deadline miss in the schedule as shown in the Figure 9(b). Therefore, we show that our approach can keep the functionally and temporally correct simulation approach on the ROS2 cyber system.

In summary, the simulation algorithm consists of offline phase and online phase. For the offline phase:

- **Generate Real Cyber System Schedule:** From the cyber system design and task parameters, we generate a real cyber system schedule for a global hyper period.
- **Construct Offline Guider:** From the generated schedule, we construct a job precedence graph called offline guider.

Note that we generate a schedule and offline guider for a global hyper period before the simulation start. After the simulation start, whenever a job simulated, next hyper period job instance is generated and update the offline guider. For online phase:

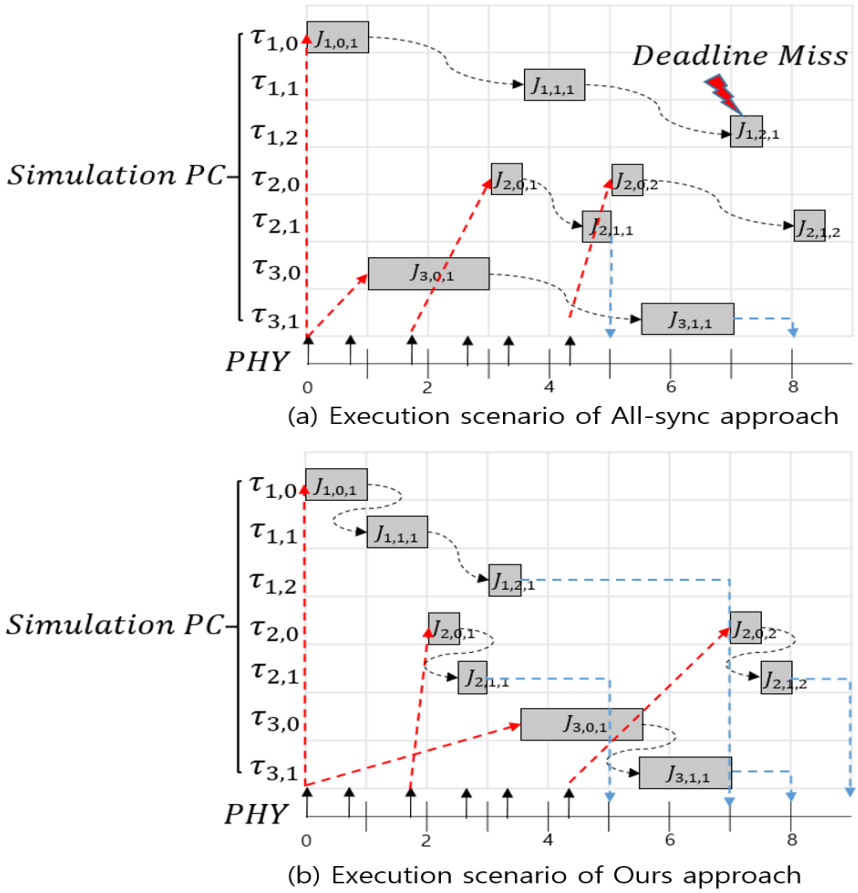


Figure 9: Swapping scheduling algorithm AllSync to Ours

- **Run Simulated Jobs:** From the offline guider, we start the simulation by scheduling jobs pushed from the offline guider with effective EDF scheduling policy.

- **Update Offline Guider and Generate Next Hyper Period Job Instance:** Whenever a job finished on the simulation PC, we generate next hyper period job instance and update the offline guider considering removed jobs and directed edges.

4 Evaluation

In this section, we explain our exeperimental setup and the results of simulation.

4.1 Experimental Setup

We conduct our simulation with 1000 synthetic workloads. We first use simple mapping function for execution time which simulation PC is 3.3 times faster than ECU. We set the period of timer callbacks with uniform [10ms, 100ms], and execution time of timer callbacks with $0.2*P_{i,0}$ and the subscribers in the same transaction follows the timers. The number of ECUs is uniform [3,10], and each ECU has tasks with uniform [1,5]. The transaction ratio is basically 50%. The read and write constrained jobs ratios are basically 30% respectively. The transaction ratio determines how many of all tasks are designated as timer callbacks. If the ratio is 0%, then there are only one timer callback so that the entire task set become one chain. The read ratio determines how many transactions can have read constrained timers. The write ratio determines how many transactions can have a write constrained subscriber. We run the simulation with Ours, AllSync, TrueTime approaches with increasing three parameters: transaction ratio, read ratio, write ratio from 0 to 100%. We measure the simulation capacity called "simulatability" by measuring "Simulatable" case of 1000 synthetic workloads. If there is a deadline miss for the schedule, then we call it "Not Simulatable". If the approach generates feasible schedule, then we call it "Simulatable".

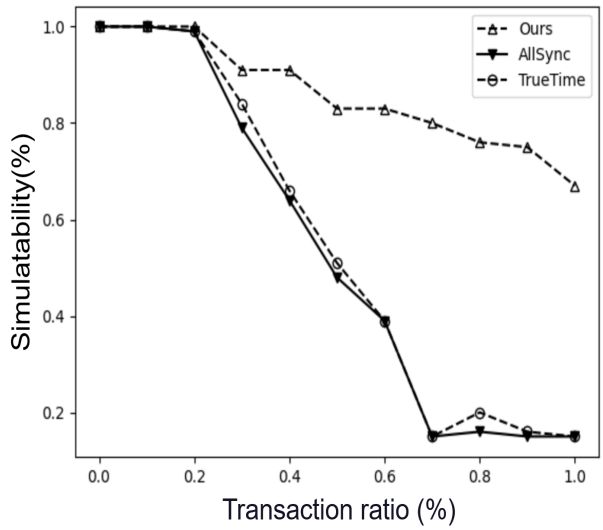


Figure 10: Simulation results of increasing transaction ratio

4.2 Simulation Results

As shown in Figure 10, we increase transaction ratio from 0 to 100%. If the transaction ratio is 0%, the number of transaction is one. In this case, all the simulation approach's simulatability is 100%. However, as the transaction ratio approaches 100%, the number of tasks and the number of transactions become the same, that is, all tasks become timer callbacks. If the number of transactions is one, the write ratio is meaningless. On the other side, if the number of transactions is the same as the number of tasks, the write ratio is critical because we assign the write constraint to the tasks 30% of the number of transactions.

For the write ratio, as shown in Figure 11, we increase the write ratio from 0 to 100%. If the write ratio is 0%, it means that the deadline for all jobs is infinite so that simulatability is 1 for every approach. However, if the write ratio is close to 100%, it is hard to simulate for AllSync and TrueTime. Because, in the case of AllSync, the

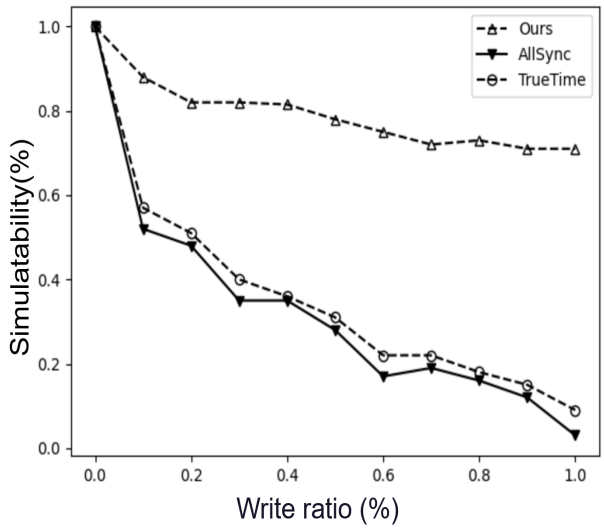


Figure 11: Simulation results of increasing write ratio

same execution order as the real cyber system, so the more write constrained jobs, the more difficult it is to finish at their actual finish times. In the case of TrueTime, due to a small freedom of scheduling, it shows a little higher simulatability than AllSync. Finally, Ours shows the highest simulatability because we run the jobs with the EDF scheduling policy.

For the read ratio, Figure 12 shows the simulation results with increasing read ratio from 0 to 100%. In this case, for the AllSync approach the read ratio is an irrelevant parameter. This is because the AllSync approach starts later than the start time for all jobs. Therefore, it has lowest simulatability. In the case of TrueTime, it shows better simulatabilty than the AllSync when the read ratio is low. However, as the read ratio becomes 100%, its simulatability getting closer to AllSync approach because its small scheduling freedom is limited to the non read constrained jobs.

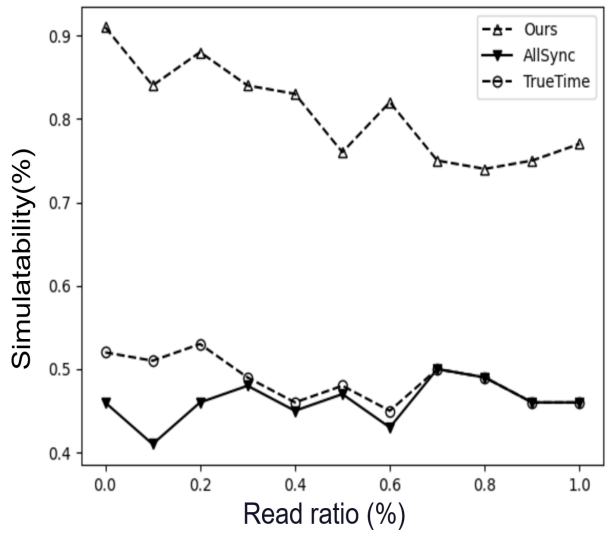


Figure 12: Simulation results of increasing read ratio

5 Conclusion

This paper proposes an extended simulation approach that can guarantee functional and temporal correctness on the ROS2 cyber system. We show the existing simulation approach still works on the ROS2 cyber system by showing the simulation results based on synthetic workloads. However, our approach still has practical issues for the future work:

- **Data Transmission Delay:** In this paper, we assume that there is no data transmission delay for the physical interactions and data exchanges between callbacks. However, we need to consider those transmission time to ensure that we can receive the same data with real cyber system from the physical system and we can transmit the same data to the physical system. In the previous work[5], the cyber system consists ECUs connected by CAN(Controller Area

Network) with TDMA bus[10] and proposed the data receive time as sum of its actual finish time, waiting time for its dedicated slot of TDMA bus and constant transmission time. However, in the ROS2 cyber system, we need to consider that ROS2 cyber system uses DDS(Data Distribution Service)[11] for the data transmissions. In the future, we plan to consider those data transmission delay.

- **Fixed Execution Time:** In this paper, we assume that all the tasks have fixed execution time for executing its function. However, if we use fixed execution time, we can not consider timing behavior caused by varying execution time. Therefore, we need to extend the approach for the varing execution time for the tasks. In the future, we expect to improve the practicality by considering varing execution time.

- **Uniprocessor:** In this paper, we assume that the simulation PC has uniprocessor so that we can keep the existing simulation approach on the ROS2 cyber system. However, in practice, there are ECUs more than 80 in the automotive systems. In the future, we plan to extend the approach to multi-core simulation based on ROS2 cyber system[12].

References

[1] Hyejin Joo, Kyoung-Soo We, Seunggon Kim, and Chang-Gun Lee. An end-to-end tool for developing cpss from design to implementation. 2016.

[2] D. Casini, T. Blaß, I. L¨utkebohle, and B. B. Brandenburg. Responsetime analysis of ros2 processing chains under reservation-based scheduling. In *Euromicro Technical Committee on Real-Time Systems (ECRTS), 2019*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[3] Kancheepuram Kattankulathur. Systematic approach in v-model development cycle for an automotive embedded control system.

[4] Simulink. *R2019a*. MathWorks Inc., Natick, Massachusetts, 2019.

[5] Kyoung-Soo We, Seunggon Kim, Wonseok Lee, and Chang-Gun Lee. Functionally and temporally correct simulation of cyber-systems for automotive systems. In *Real-Time Systems Symposium (RTSS), 2017 IEEE*, pages 68–79. IEEE, 2017.

[6] ROS Overview. `http://wiki.ros.org/ROS/Introduction`. 2020.

[7] ROS2 Overview. `https://index.ros.org/doc/ros2/`. 2020.

[8] Chung Laung Liu and James W Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.

[9] M. Spuri and J. A. Stankovic. How to integrate precedence constraints and shared resources in real-time scheduling. In *Transactions on Computers (TC), 1994*, pages 1407–1412. IEEE, 1994.

[10] Thomas Fuhrer. Time triggered communication on can (time triggered can-ttcan). In *Proceedings 7th International CAN Conference, 2000*, 2000.

[11] Gerardo Pardo-Castellote. Omg data-distribution service: Architectural overview. In *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, pages 200–206. IEEE, 2003.

[12] Wonseok Lee, Jaehwan Jeong, Seonghyeon Park, and Chang-Gun Lee. Practical multicore extension of functionally and temporally correct real-time simulation for automotive systems. In *Cyber Physical Systems. Model-Based Design*, pages 127–152. Springer, 2019.

요약(국문초록)

본 논문은 ROS2를 기반으로 설계된 자동차 사이버 시스템을 기능적/시간적으로 정확하게 시뮬레이션하는 방법론을 제안한다. 앞선 연구에서는 기존의 시뮬레이션 기법들의 기능적인 정확성만 보장하는 문제에서 발생하는 한계점을 극복하고 동시에 효율적으로 작업을 수행하는 시뮬레이션 기법이 제안되었다. ROS2 기반의 자동차 사이버 시스템에서는 기능 수행의 행태가 기존의 자동차 사이버 시스템과는 다르다는 것을 인지하여 본 논문에서는 앞선 연구에서 제안하는 시뮬레이션 기법의 핵심 아이디어가 유지되면서 ROS2 기반의 자동차 사이버 시스템에 적용이 될 수 있도록 시뮬레이션 기법을 제안한다. 제안하는 방법에서는 ROS2 스케줄링을 고려한 시스템 모델을 정의하고 이를 기반으로 실제 사이버 시스템의 스케줄을 예측하고 선행 관계 그래프를 생성하여 기존의 시뮬레이션 기법이 그대로 적용될 수 있도록 한다. 제안하는 방법은 임의적으로 생성된 워크로드를 통해 다른 시뮬레이션 알고리즘과 함께 시뮬레이션 용량을 측정하고, 제안하는 방법이 싱글코어 시뮬레이터에서 가장 높은 시뮬레이션 용량을 가지는 것을 보인다. 따라서, 기존의 기능적/시간적으로 정확한 시뮬레이션 기법이 ROS2 기반의 자동차 사이버 시스템에서도 적용이 가능하며, 이를 활용하여 기존의 자동차 사이버 시스템뿐만 아니라 ROS2 기반의 자동차 사이버 시스템을 효과적으로 시뮬레이션 할 수 있다.

주요어 : 자동차 시스템, 실시간 시뮬레이션, ROS2 프레임워크
학 번 : 2019-23556