GNSS-Based System Time Synchronization Mechanism for Cyber-Physical Systems

Kazuya Harayama, Takanori Yokoyama, and Myungryun Yoo
Department of Computer Science, Tokyo City University, Setagaya-ku, Tokyo 158-8557 Japan
Email: itm85hk@gmail.com; {tyoko; myoo}@tcu.ac.jp

***Abstract*—The paper presents a system time synchronization mechanism of a Real-Time Operating System (RTOS) for loosely coupled distributed Cyber-physical Systems (CPS) such as distributed control systems connected with wireless networks or wide-area networks. In CPS, computations directly interact with the physical world based on the physical time. Meanwhile, application tasks that perform computation are managed by a RTOS according to the system time. Thus, a precisely synchronized system time is required for high performance control of the physical world. The system time synchronization mechanism presented in this paper provides the system time that is precisely synchronized with UTC (coordinated universal time). The system time synchronization mechanism compensates tick rate, tick phase and the value of the system time utilizing the Pulse Per Second (PPS) signal of a GNSS (global navigation satellite systems) receiver module. We build the system time synchronization mechanism into a RTOS, with which embedded computers synchronously execute application tasks according to the precisely synchronized system time. The evaluation results show that the synchronization error is sufficiently small for practical embedded control systems. The system time synchronization mechanism can be easily applied to other RTOS.**

***Index Terms*—System time synchronization, real-time operating systems, cyber-physical systems, distributed control systems, GNSS**

I. Introduction

Cyber-physical Systems (CPS) are real-time systems, in which computations directly interact with the physical world where time cannot be abstracted away [1]. Application tasks are managed by Real-Time Operating Systems (RTOS) according to the system time in most CPS. Distributed embedded control systems such as automotive control systems, multi-vehicle control systems and avionics systems are hard real-time CPS, in which the delay and jitter introduced by the computer system may lead to significant performance degradation [2]. A precisely synchronized system time is required for high performance control of the physical world.

NTP (network time protocol) [3] is widely used to synchronize the system times of nodes in distributed computing systems. However, NTP is not sufficient for hard real-time CPS because the accuracy of time synchronization is not enough for a distributed control system to correctly control the physical world. The IEEE 1588 PTP (precision time protocol) is more accurate time synchronization protocol. Cochran *et al.* presented a method to synchronize the Linux system time with a PTP hardware clock [4]. A time-triggered architecture with clock synchronization is useful for hard real-time distributed systems [5]. Chiba *et al.* presented a distributed operating system with a global time mechanism [6], which is supported by the clock synchronization of FlexRay [7]. However, these systems require wired local-area networks and are not applicable for loosely-coupled CPS such as distributed control systems connected with wireless networks or wide-area networks.

GNSS (global navigation satellite systems) such as GPS (global positioning system) and GLONASS (globalnaya navigatsionnaya sputnikovaya sistema) are useful for time transfer [8]. Bogdanov *et al.* showed that GNSS time offsets are maintained within ±20 ns [9]. If each computer has a GNSS receiver module, the system time of a distributed CPS can be precisely synchronized without wired networks. Kim *et al.* presented a clock synchronization module using GPS for a middleware called TMOSM (time-triggered message-triggered object support middleware) for wide-area distributed systems [10]. Quesada *et al.* presented and evaluated GPS-based clock synchronization methods using a PPS (pulse per second) signal of the GPS receiver module [11]. Kubczak *et al.* presented preprocessing for fast synchronization of high-stability oscillators disciplined by the GNSS PPS signal [12]. Hollós *et al.* presented a GNSS-based reference clock connection interface for IEEE 1588 master clocks [13]. Hasan *et al.* demonstrated that the coverage availability and the timing accuracy of GNSS-based time synchronization are acceptable for most vehicular ad-hoc networks [14]. However, these systems are not for the synchronization of the system time of RTOS, which is used for task scheduling.

We already presented a RTOS that provides a system time, which is precisely synchronized based on GNSS [15]. However, the system time synchronization mechanism of the RTOS requires a GNSS module that provides not only a PPS signal but also a low jitter high rate clock signal. There are a few modules that provide low jitter high rate clock signal and they are expensive.

Manuscript received August 26, 2019; revised November 5, 2019; accepted December 30, 2019.

Corresponding author: Takanori Yokoyama (email: tyoko@tcu.ac.jp).

Most GNSS receiver modules provide just a PPS signal, which provides sufficient accuracy for hard real-time embedded systems as Niu *et al*. reported in [16].

This paper presents a GNSS-based system time synchronization mechanism that requires just the PPS signal of a GNSS receiver module. The system time of a RTOS is represented by a counter, the unit of which is called a *tick*. The system time counter is updated by an interrupt, which is periodically issued by a hardware timer. The GNSS-based system time synchronization mechanism is implemented as an interrupt routine and synchronizes the system time counter with UTC (coordinated universal time) by compensating the rate of the tick, the phase of the tick and the value of the system time referring to the PPS signal. We build the system time synchronization mechanism into a RTOS, which manages application tasks according to the precisely synchronized system time. By using the RTOS, we can develop a hard real-time CPS, in which coherent task scheduling is performed.

The rest of the paper is organized as follows. Section II presents system time synchronization utilizing the PPS signal of a GNSS receiver module. Section III describes the implementation of the system time synchronization mechanism. Section IV evaluates the performance of the system time synchronization mechanism and Section V concludes the paper.

II. GNSS-Based System Time Synchronization

A. Design Policy

Fig. 1 illustrates an example CPS, in which embedded computers cooperatively control physical objects based on a GNSS-based synchronized system time. The system time on each embedded computer is synchronized with UTC referring to a GNSS receiver module signal on each embedded computer. Thus, embedded computers can synchronously execute application tasks according to the synchronized system time.

The design policies of the GNSS-based system time synchronization mechanism are shown below.

- Special hardware should not be used.
- The major modification of the RTOS is not required.

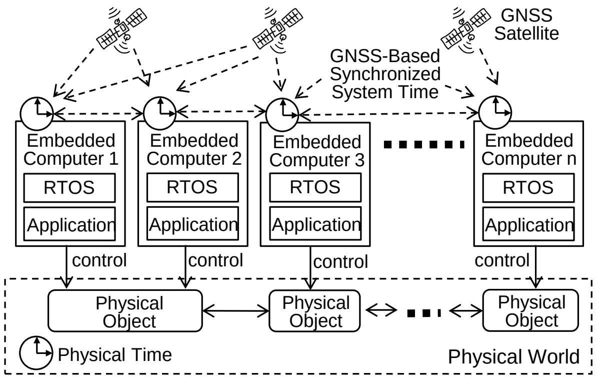

Fig. 1. Cyber-physical system with GNSS-based synchronized system time.

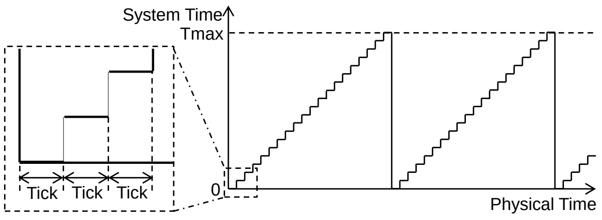

Fig. 2. Tick and system time.

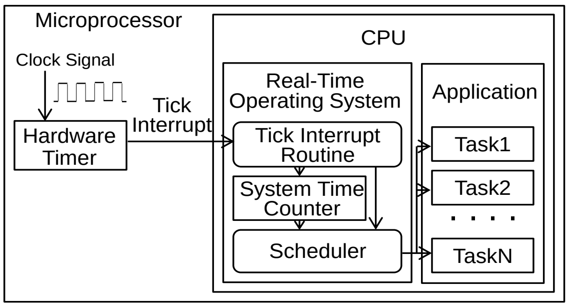

Fig. 3. System time updating mechanism.

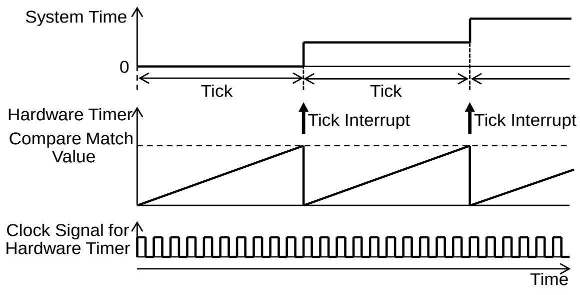

Fig. 4. Tick Interrupt

The first policy is to prevent increasing the cost. We implement the system time synchronization mechanism by software on a general-purpose microprocessor with a GNSS receiver module. The second policy is to easily build the system time synchronization mechanism into various RTOS.

Fig. 2 illustrates the tick and the system time. A tick is a unit of the system time. T_{max} means the maximum value of the system time. If the value of the system time becomes to be equal to T_{max}, the value is reset to zero at the next tick.

Fig. 3 illustrates the system time updating mechanism. The hardware timer counts the clock signal. The system time is represented by the system time counter of the RTOS. The system time counter is periodically updated by the tick interrupt routine, which is issued by the hardware timer interrupt. The scheduler of the RTOS manages the application tasks according to the system time.

Fig. 4 illustrates the relation of the system time and the hardware timer. A compare match timer is usually used to issue the tick interrupt. When the value of the hardware timer becomes to be equal to the compare match value, the tick interrupt is issued and then the value is reset to zero. The compare match value is set so that the tick interrupt occurs in each tick.

We present a system time synchronization mechanism that adjusts the compare match value of the hardware timer to compensate the tick and the system time.

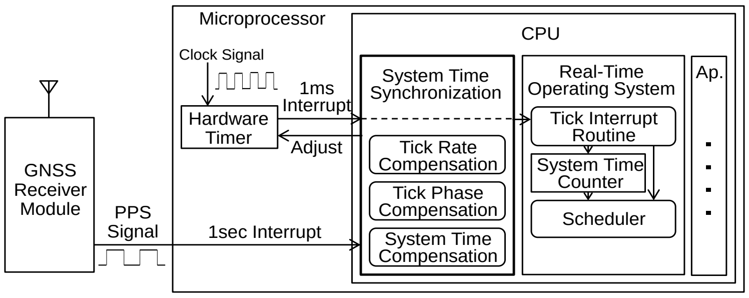

Fig. 5. System time synchronization mechanism

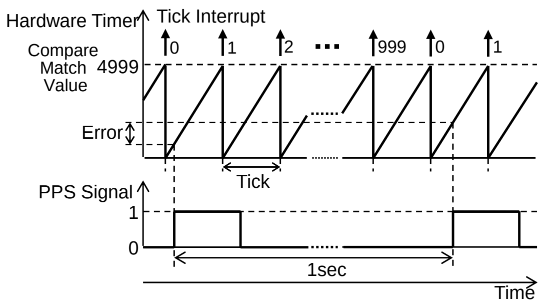

Fig. 6. Tick rate error.

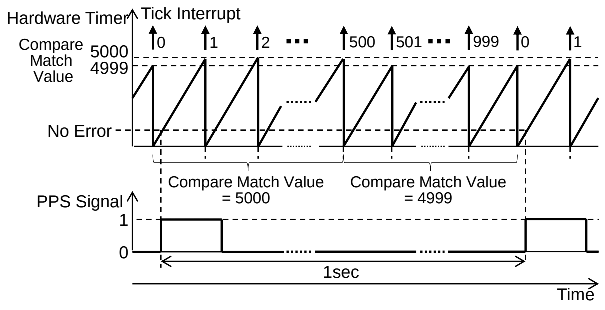

Fig. 7. Tick rate compensation.

B. Synchronization Mechanism

To precisely synchronize the system time, the value of the system time must be kept identical at any instant. Thus, not only the value of the system time at an instant but also the tick rate and the tick phase must be identical. The system time synchronization mechanism compensates the tick rate, the tick phase and the value of the system time. The compensations are performed by adjusting the compare match value of the hardware timer referring to the PPS signal of a GNSS receiver module.

Fig. 5 illustrates the time synchronization mechanism, which compensates the tick rate, the tick phase and the value of the system time just by using the PPS signal. The details of the compensation are described in Section II-C and Section II-D

The GNSS receiver module outputs the signals under the condition that sufficient GPS satellites are captured. If sufficient GPS satellites are not captured, the GNSS receiver module stops outputting the PPS signal. So the tick synchronization is performed so long as the GNSS receiver module outputs the PPS signal. If the GNSS receiver module resumes outputting the PPS signal, the tick synchronization also resumes. The error of the system time is small and can be compensated in a short time if the PPS signal suspended duration is not so long.

C. Tick Rate Compensation

The clock signal is generally generated by a crystal oscillator, which usually has frequency deviation. The clock rate also varies depending on the temperature. The deviation and the variation cause tick rate error. Fig. 6 illustrates the hardware timer for the tick interrupt and the PPS signal in the case that the tick is 1ms and the frequency of the clock signal is 5MHz. In this case, the compare match value is set to 4999 and the tick interrupt is issued 1000 times per second. The figure also shows the tick rate error in one second, which is a difference between the values of the hardware timer at the timing of the rising edge of the PPS signal.

We present a method to compensate the tick rate by precisely adjusting the compare match value of the hardware timer in each one second interval. Fig. 7 illustrates the hardware timer and the PPS signal. The tick rate is compensated by precisely adjusting the compare match value. In this example, the compare match value is set to 5000 in the first half duration and set to 4999 in the latter half duration in one second. The durations are calculated by the system time synchronization mechanism to minimize the tick rate error. This makes it possible to keep the tick rate error under the period of the clock signal without using a low jitter high rate clock signal.

D. Tick Phase and System Time Compensation

The tick phase and the value of the system time are compensated as same as the previous system time synchronization mechanism presented in [15].

Fig. 8 illustrates the PPS signal, the hardware timer and the system time. The timing to update the system time must be matched to the timing of the rising edge of

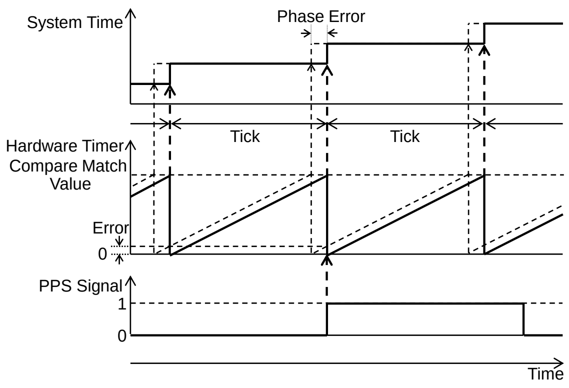

Fig. 8. Tick phase compensation.

the PPS signal. The solid line of the value of the hardware timer and the solid line of the value of the system time show a case in which no phase error exists and the broken lines show a case in which a phase error exists.

The tick phase is compensated by adjusting the tick interval, i.e., by adjusting the compare match value of the hardware timer until the timing of updating the system time becomes matched to the timing of the rising edge of the PPS signal.

The system time is correctly maintained just by the tick rate compensation and tick phase compensation so long as the GNSS receiver module outputs the PPS signal. However, if the GNSS receiver module stops outputting the PPS signal, the system time may not be correctly maintained. If the value of the system time is not correct when the GNSS receiver module resumes outputting the PPS signal, the system time must be compensated.

Fig. 9 illustrates the PPS signal and the system time in the case that the maximum value of the system time (T_{max}) is one second. The value of the system time must be just zero at the timing of the rising edge of the PPS signal in this case.

The solid line of the value of the system time shows a case in which no system time error exists and the broken line shows a case in which a system time error exists. The system time is compensated by adjusting the tick interval, i.e., by adjusting the compare match value of the hardware timer until the system time becomes just zero, one or multiple seconds at the timing of the rising edge of the PPS signal.

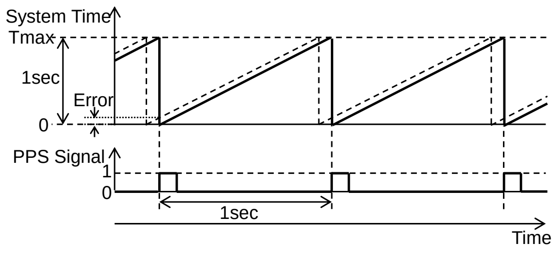

Fig. 9. System time compensation.

III. Implementation

A. Hardware and Operating System

We use an evaluation board in which a microprocessor called H8S/2638F is installed. The clock rate of the microprocessor is 20MHz. We also use a GNSS receiver module called GT-87 [17], which supports GPS, GLONASS, SBAS and QZSS. The accuracy of the PPS signal is less than 15ns(1σ)(@-130 dBm) and less than 50ns(1σ) (@-150 dBm). Fig. 10 illustrates the microprocessor and the GNSS receiver module. The hardware timer *Timer 0* connected with the *f*/4 signal generated by dividing the clock by 4 is used to issue an interrupt in each 1ms interval. We also use another hardware timer *Timer 2* connected to the PPS signal to issue an interrupt in each one second interval.

We build the system time synchronization mechanism into an OSEK OS [18] called TOPPERS/ATK1, which is developed by TOPPERS project [19]. OSEK OS provides system services for task management, alarms, events, resources and interrupt management. The counter and alarm mechanism is used to implement periodic tasks in OSEK OS. The system time counter is a counter to represent the system time. An alarm is used to activate a task, set an event or call an alarm call back routine. Periodic tasks are activated by alarms connected with the system time counter.

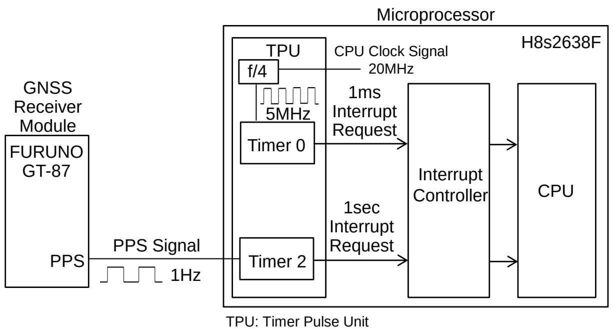

Fig. 10. Microprocessor and GNSS receiver module.

Fig. 11 illustrates the hardware timer and the *tick interrupt routine* of the original TOPPERS/ATK1 for H8S/2638F. The default tick is 1ms, so the compare match value of *Timer 0* is 4999. The default maximum value of the system time is 1000 (999 to be exact), i.e., one second. *Tick interrupt routine* is issued by *Timer 0* and executes the processing for the system time counter and the alarms. *Tick interrupt routine* of TOPPERS/ATK1 is implemented as an ISR (interrupt service routine) of Category 2, which can use OSEK OS system services.

B. System Time Synchronization Mechanism

Fig. 12 illustrates hardware timers and interrupts for the system time synchronization mechanism. *1ms interrupt routine* issued by *Timer 0* in each 1ms interval and *PPS interrupt routine* issued by *Timer 2* in each one second interval are added. As shown by the figure, the system time synchronization mechanism, which consists of *1ms interrupt routine* and *PPS interrupt routine*, can be added without modifying the program of the RTOS including *tick interrupt routine*, except for the hardware timer setup code. Thus, the mechanism can be easily applied to other RTOS.

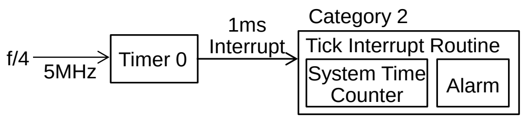

Fig. 11. ISR structure of original RTOS.

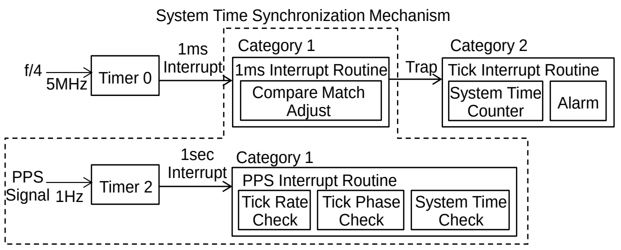

Fig. 12. ISR structure of system time synchronization mechanism.

Tick interrupt routine is activated by a trap (software interrupt) issued by *1ms interrupt routine*. *PPS interrupt routine* and *1ms interrupt routine* are ISRs of Category 1, which cannot use OSEK OS system services, because the overhead of an ISR of Category 1 is less than the overhead of an ISR of Category 2. The priority of an ISR of Category 1 is higher than the priority of an ISR of Category 2. The priority of *PPS interrupt routine* is higher than the priority of *1ms interrupt routine*.

PPS interrupt routine checks the tick rate as described in Section II-C. *PPS interrupt routine* also checks the tick phase and the value of the system time as described in Section II-D. If the tick rate error or the tick phase error is detected to be greater than their thresholds or the system time error is detected, *1ms Interrupt Routine* adjusts the compare match value of *Timer 0* for compensation. The threshold of the tick rate error is 1 µs (0.1% of the tick interval) and the threshold of the tick phase error is 5 µs (0.5% of the tick interval). The parameters for the compensation are decided so that the variation of the compare match value is less than 1%.

IV. Experimental Evaluation

A. Overhead of System Time Synchronization

We have measured the CPU execution times of *PPS interrupt routine*, *1ms interrupt routine* and *tick interrupt routine*. Table I shows the average value and the maximum value (written in parentheses) of each interrupt of the RTOS with the system time synchronization mechanism presented in this paper. The execution time of *tick interrupt routine* is shown separately in the case that no alarm is defined (*without alarm*), in the case that an alarm is defined and no task is activated (*with alarm*), and in the case that an alarm is defined and a task is activated by the alarm (*task activation*). The table also shows the CPU execution times of the RTOS developed in the previous work [15] that utilizes not only the PPS signal but also a low jitter high rate clock signal for comparison. The column *presented* shows the CPU execution time of the system time synchronization mechanism presented in this paper and the column *previous* shows the CPU execution time of the previous work.

The CPU execution time of *1ms interrupt routine* of the presented mechanism is much less than the previous work. This is because the tick rate check, which is executed by *1ms interrupt routine* in the previous work, is executed by *PPS interrupt routine* in the presented mechanism. The execution time of *PPS interrupt routine* of the presented mechanism is more than the previous work. However, *1ms interrupt routine* is executed a thousand times in a second and *PPS interrupt routine* is executed once in a second. Thus, the overhead of the presented mechanism is much less than the previous work.

The CPU execution time of *1ms interrupt routine* of the presented mechanism is also much less than the execution time of *tick interrupt routine*, which is the original program of TOPPERS/ATK1 and not modified. Thus, we think that the overhead of the presented mechanism is acceptable for practically embedded control systems.

TABLE I. CPU EXECUTION TIME OF INTERRUPTS

Interrupt Routine		CPU Execution Time [µs]	
		Presented ave.(max.)	Previous ave.(max.)
PPS		13.3 (16.8)	10.6 (10.6)
1ms		2.9 (2.9)	9.7 (10.5)
Tick	without alarm	10.1 (10.1)	10.1 (10.1)
	with alarm	13.1 (13.1)	13.1 (13.1)
	task activation	45.8 (45.8)	45.8 (45.8)

B. Accuracy of System Time Synchronization

To evaluate the accuracy of the system time synchronization, we have measured the synchronization error by comparing the compare match timing of the hardware timers of different microprocessors. We have set the hardware timer parameters so that the output of the hardware timer is toggled by compare match and have measured the time difference between the toggle outputs of the hardware timers of different microprocessors using an oscilloscope. The time difference means the synchronization error of the system time.

The maximum time difference is 10 µs in the case of the system time synchronization mechanism presented in this paper and is 18 µs in the case of the system time synchronization mechanism of the previous work. We think that the tick rate compensation by the presented mechanism is more stable than the tick rate compensation of the previous work, which is executed in each tick referring to the low jitter high rate clock signal.

The synchronization error of the presented mechanism is less than 10 µs. The error is much less than the CPU execution time of the tick interrupt with task activation (45.8 µs as shown in Table I), which means the task activation time. It is also less than 1% of 1ms, which is the typical tick time of RTOS used in embedded control systems. Thus, we think that the synchronization error is sufficiently small for practical embedded control systems.

V. Conclusion

We have presented a GNSS-based system time synchronization mechanism that utilizes just the PPS signal of a GNSS receiver module. The system time is synchronized with UTC by compensating the tick rate, the tick phase and the value of the system time referring to the PPS signal.

We have built the system time synchronization mechanism into an OSEK OS called TOPPERS/ATK1 and evaluated the overhead of the mechanism and the accuracy of the system time synchronization. The evaluation results show that the overhead is acceptable and the synchronization error is sufficiently small for practical embedded control systems.

The system time synchronization mechanism issues the tick interrupt that updates the system time of the RTOS. The mechanism is added without modifying the source code of the RTOS except for the hardware timer setup code, so it can be easily applied to other RTOS.

A time-triggered operating system such as OSEKtime [20] may be needed for strict hard real-time applications that require less jitter. The future work is to develop a time-triggered operating system with GNSS-based system time synchronization.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Kazuya Harayama carried out the implementation and the experimental evaluation. Takanori Yokoyama conducted the research and wrote the paper with the cooperation of Myungryun Yoo. All authors had approved the final version.

ACKNOWLEDGMENT

The authors would like to thank the developers of TOPPERS/ATK1. This work was supported by JSPS KAKENHI Grant Number 18K11225.

REFERENCES

[1] E. A. Lee, "Cyber physical systems: Design challenges," in *Proc. 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, Orlando, 2008, pp. 363-369.

[2] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K. Arzen, "How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime," *IEEE Control Systems*, vol. 23, no. 3, pp. 16-30, May 2003.

[3] D. L. Mills, "Internet time synchronization: the network time protocol," *IEEE Trans. on Communications*, vol. 39, no. 10, pp. 1482-1493, Oct. 1991.

[4] R. Cochran, C. Marinescu, and C. Riesch, "Synchronizing the Linux system time to a PTP hardware clock," in *Proc. IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, Munich, 2011, pp. 87-92.

[5] H. Kopetz, "Should responsive systems be event-triggered or time-triggered?" *IEICE Transaction on Information & Systems*, vol. E76-D, no. 11, pp. 1325-1332, Nov. 1993.

[6] T. Chiba, Y. Itami, M. Yoo, and T. Yokoyama, "A distributed real-time operating system with location-transparent system calls for task management and inter-task synchronization," in *Proc. IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Changsha, 2011, pp. 1133-1138.

[7] R. Makowitz and C. Temple, "FlexRay - a communication network for automotive control systems," in *Proc. IEEE International Workshop on Factory Communication Systems*, Torino, 2006, pp. 207-212.

[8] W. Lewandowski, J. Azoubib, and W. J. Klepczynski, "GPS: Primary tool for time transfer," *Proceedings of the IEEE*, vol. 87, no. 1, pp. 163-172, Jan. 1999.

[9] R. P. Bogdanov, A. Druzhin, T. Primakina, and A. Tiuliakov, "Analysis of GNSS time scales," in *Proc. European Frequency and Time Forum*, Turin, 2018, pp. 181-184.

[10] K. H. Kim and S. F. Jenks, "The TMO scheme for wide-area distributed real-time computing and distributed time-triggered simulation," in *Proc. IEEE International Parallel and Distributed Processing Symposium*, Rome, 2007, pp. 1-6.

[11] J. Quesada, J. U. Llano, R. Sebastian, M. Castro, and E. Jacob, "Evaluation of clock synchronization methods for measurement and control using embedded Linux SBCs," in *Proc. 9th International Conference on Remote Engineering and Virtual Instrumentation (REV)*, Bilbao, 2012, pp. 1-7.

[12] P. Kubczak, M. Kasznia, and M. Jessa, "Preprocessing for fast synchronization of high-stability oscillators disciplined by GNSS 1 PPS signal," in *Proc. European Frequency and Time Forum*, Turin, 2018, pp. 234-239.

[13] Á. E. Hollós and T. Kovácsházy, "Improved reference clock connection interface for prototype IEEE 1588 master clocks," in *Proc. 19th International Carpathian Control Conferences*, Szilvasvarad, 2018, pp. 371-376.

[14] K. F. Hasan, Y. Feng, and Y. C. Tian, "GNSS time synchronization in vehicular ad-hoc networks: Benefits and feasibility," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3915-3924, Dec. 2018.

[15] T. Yokoyama, A. Matsubara, and M. Yoo, "A real-time operating system with GNSS-based tick synchronization," in *Proc. 3rd IEEE International Conference on Cyber-Physical Systems, Networks, and Applications*, Hong Kong, 2015, pp. 19-24.

[16] X. Niu, K. Yan, T. Zhang, Q. Zhang, H. Zhang, and J. Liu, "Quality evaluation of the Pulse Per Second (PPS) signals from commercial GNSS receivers," *GPS Solutions*, vol. 19, no. 1, pp. 141-150, Jan. 2015.

[17] Furuno Electric Co., Ltd. Timing Multi-GNSS Receiver Module Model GT-87. [Online]. Available: https://www.furuno.com/en/products/gnss-module/GT-87

[18] TOPPERS Project. TOPPERS/ATK1. [Online]. Available: http://www.toppers.jp/atk1.html

[19] OSEK/VDX, *Operating System*, Version 2.2.3, 2005.

[20] OSEK/VDX, *Time-Triggered Operating System*, Version 1.0, 2001.

Kazuya Harayama received his B.E. degree from Tokyo City University in 2019. His interest includes embedded systems and software engineering.

Takanori Yokoyama received his B.E. degree, M.E. Ph.D. degrees in information science from Tohoku University in 1981, 1983, and 2002 respectively. He joined Hitachi, Ltd. in 1983. He joined Musashi Institute of Technology in 2004. He is now a professor of Tokyo City University. His research interest includes embedded systems, distributed systems and software engineering. He is a member of IEEE, ACM, IPSJ and IEICE.

Myungryun Yoo received B.E. degree from Andong National University, Korea in 1994, M.S. degree from Pohang University of Science & Technology, Korea in 1996 and Ph.D. degree from Graduate School of Information, Production & Systems, Waseda University, Japan in 2006. She joined Musashi Institute of Technology in 2006. She is now a professor of Tokyo City University. Her research field is Real-Time System, Scheduling, Multimedia System, etc.