2016 IEEE International Conference on Computational Science and Engineering, IEEE International Conference on Embedded and Ubiquitous Computing, and International Symposium on Distributed Computing and Applications to Business, Engineering and Science

Relative Time Synchronization of Distributed Applications for Software-in-the-loop Simulation

Sunghee Lee, Bueng Il Hwang, Kang-Bok Seo, Woo Jin Lee*
School of CSE
Kyungpook National University
Daegu, Republic of Korea
lee3229910@gmail.com, halove200@gmail.com, seokang13@naver.com, woojin@knu.ac.kr*

***Abstract*—As distributed systems such as automotive, medical, manufacturing automation become larger and more complex, it is difficult to test these systems. Also, the synchronization of distributed applications make the testing more difficult. In the Software-in-the-Loop (SiL) simulation, a synchronization method among clock of applications is provided for virtual hardware devices and environment. A typical synchronization technique is that a single global clock synchronizes local clocks of other sites but may make high accuracy but low performance in SiL simulation because synchronization occurs in all nodes when the global clock ticks. Also there are some issues of which node should be selected, how much time the global clock drifts when adding new sites, etc. In the paper, we propose a method of clock synchronization based on relative time to enhance the simulation performance.**

Keywords—Synchronization; Relative; Distributed; Simulation; Software-in-the-loop;

I. INTRODUCTION

Recently, as the IT convergence industry has been enlarged and more wider, the scale of embedded system become more bigger in the form of distributed system such as automotive, medical, manufacturing automation, etc. [1-3]. Thus, it is necessary to test applications of these distributed embedded systems but it is difficult to test these applications since the testing environment of distributed applications should provide both the execution environment and synchronization mechanism among distributed sites [3-5]. The process of testing embedded testing is shown Fig. 1. Generally, testing of embedded system is performed with MiL(Model-in-the-Loop), SiL(Software-in-the-Loop) and HiL (Hardware-in-the-Loop) in order. In this paper, we consider only level of SiL [6] which is testing without hardware. In SiL simulation of distributed applications, however, time synchronization is essential. In wireless sensor network, for example, time synchronization is a major topic because of high accuracy [7].

In distributed applications there are two issue of time synchronization. One is synchronization between the clock in an application and the clock in real world, which is absolute synchronization. The other is synchronization among clock of applications, which is relative synchronization [5]. In SiL simulation relative synchronization is important more than absolute synchronization because environment and hardware is not real but virtual. Thus, in this paper, we consider only relative synchronization. A typical synchronization technique is using global clock [8]. This technique is that only global clock synchronizes clocks of the others but may make high accuracy low performance in SIL simulation because synchronization occurs in all nodes when global clock ticks. Also there are some issues that which node should be selected, that new node be required, that how much time global clock drift, etc.

So for high performance of SiL simulation including in synchronization and above issues, we propose a method of relative time synchronization. This paper is organized as follows: Section II introduces synchronization of related works; Section III presents the main idea of relative time synchronization; Section IV shows case study; Section V shows evaluation of our method; Section VI provides conclusion.

II. REALATED WORKS

IEEE 1588 standard [9-11] defines PTP(precision time protocol) which enables precise synchronization of clocks in distributed system. Because each system has its clock which is

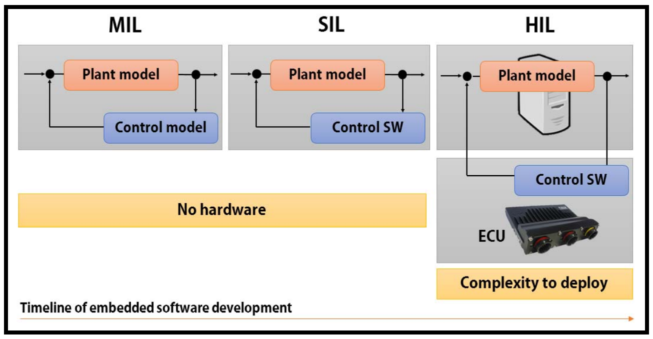

Fig. 1. Process of testing embedded system

*corresponding author

different to others, there is a single global clock which should be absolute basis of time and lead local clocks in distributed system as shown in Fig 2. All local clock is just synchronized by global clock. In process of this synchronization local clock is calculated by below equations as shown in Fig. 3 and as in [9], there is the details such as sequence of synchronization and related message.

Main issue is decision on cycle of global clock. Consequently, cycle of global clock cannot exceed the greatest common denominator among control cycle of distributed systems. Because all distributed system is synchronized by global clock, cycle of global clock encounters cycle of all local clock. In addition, in order to synchronize local clock with global clock, it is necessary to calculate equations such as (1), (2), (3), (4) and (5). Local clock is accuracy synchronized by (5). But this process may get much overhead. For example, let's assume that global clock ticks per 10ms, control cycle of system A is 20ms and control cycle of system B is 50ms, synchronization continuously occurs when global clock ticks per its cycle. As a result, synchronization should occurs 2 times in order that control cycle of system A operates once. Similarly, synchronization should occurs 5 times in order that control cycle of system B operates once. In the point of view of distributed systems, synchronization is very inefficient process to operate. The shorter cycle of global clock is, the more overhead of synchronization increases. In the worst case the cycle of global clock is extremely short, overhead of synchronization is divergence. Also the higher absolute accuracy of synchronization is, the more overhead of synchronization is.

III. A METHOD OF RELATIVE SYNCHRONIZATION IN DISTRIBUTED APPLICATION

A. *Structure of relative synchronization in SIL simulation*

The absolute accuracy of synchronization is very important in real distributed environment whereas relative accuracy of synchronization is important in SIL simulation. The relative accuracy of synchronization means sequence of operation of distributed systems. Each distributed system gathers data from others and operates at the same time. When a distributed system operate, the distributed system need data which other distributed systems has gathered just before. For example, assume that system A is gas sensing system such that its control cycle is 10ms and system B is alarming system such that its control cycle is 50ms. When global clock become 50ms, system A senses and system B alarms in case of sensing gas concurrently. In this situation, system B uses data from system A not which has gathered at the moment that global clock become 50ms but which has gathered at the moment global clock become 40ms.

This means that synchronization occurs 5 times with existing synchronization but occurs once with relative accuracy. In this paper, we propose a method of synchronization using these relative accuracy. And synchronization should considers some delay, jitter, and error in real environment but we consider only method of synchronization since these factors cannot affect method of synchronization. We propose a method of relative synchronization and consider only method of synchronization without many factors such as delay, jitter, error, etc. These factors affect both absolute synchronization and relative synchronization. So we don't care these factors.

When SIL simulation of distributed applications is performed, there are sensor applications, actuator applications and global clock as shown in Fig. 4. Each application has local clock for synchronization. Before simulation, first of all, it is essential to classify 2 group such as sensor node and actuator node since all actuator node operates data which sensor nodes has gathered just before at the moment. So the shortest control cycle of actuator node is determined as cycle of global clock. This classification determine cycle of global clock and then the cycle of global clock should be initialized for synchronization. Fig. 5 shows initializing process.

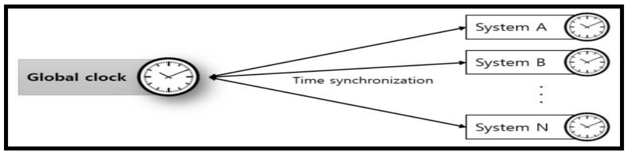

Fig. 2. Time synchronization using global clock

$$t_{offset2global} = t_{sync\_recv} - t_{global\_timestamp} - t_{onway\_delay} \quad (1)$$

$$t_{onway\_delay} = (t_{dglobal2local} - t_{dlocal2global}) / 2 \quad (2)$$

$$t_{dglobal2local} = t_{sync\_recv} - t_{global\_timestamp} \quad (3)$$

$$t_{dlocal2global} = t_{delay\_req\_recv} - t_{delay\_req\_send} \quad (4)$$

$$T_{local} = T_{local} + t_{offset2global} \quad (5)$$

where, T_{local} is the local clock offset to the global clock
t_{sync_recv} is the time at receiving *sync* from global clock
$t_{global\_timestamp}$ is the time at sending *sync* to local clock
$t_{delay\_req\_recv}$ is the time at receiving the *delay_req* from local clock
$t_{delay\_req\_send}$ is the time at sending the *delay_req* to global clock

Fig. 3. Equations to calculate local clock for time synchronization

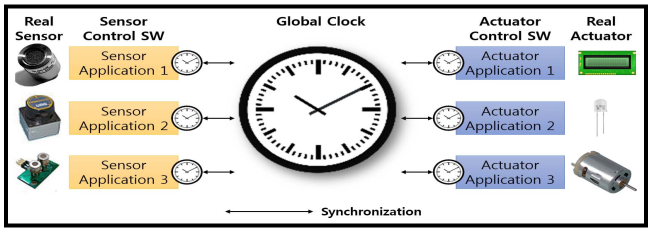

Fig. 4. Outline of synchronization in SIL simulation

```
Create all applications;
Create global clock process;

Classify all applications into sensor, actuator;
if (Application$_i$ is both sensor and actuator)
        Application$_i$ is classified actuator;
End if
Calculate greatest common denominator among cycle of all actuator applications;
Set calculated greatest common denominator value to cycle of global clock;

Run applications;
Make global clock start to tick;
```

Fig. 5. Pseudo code of initializing part before SIL simulation start

B. Synchronization Protocol

IEEE 1588 synchronizing need *Sync, Follow_Up, Delay_Req, Delay_resp* messages but our protocol need just *Tick* and *Ticked* message for handshaking. In SIL simulation no delay calculation exist since just relative synchronization is needed. *Tick* message is sent to local clock by global clock with its clock. *Ticked* message is sent to global clock by local clock with identifier of the application, which means local clock is synchronized.

C. Synchronization in SIL simulation

For synchronization in the point of view of distributed application each control loop of application is waiting for *tick* message from global clock to check time of global clock and then if the time of global clock is equal to or greater than next time of local clock, control loop is executed when local clock receives *tick* message. After an execution, local clock sends global clock *ticked* message. Applications repeats these process during simulation as shown in Fig. 6. All distributed application always cannot help getting data from other applications which has gathered just before at the moment. Thus, this relative synchronization is possible.

In the point of view of global clock it should see all distributed application is ticked which *ticked* messages is sent by all distributed application. That all distributed application is ticked means all distributed application become synchronized. In previous initializing part the shortest control cycle of actuator node is determined as cycle of global clock. Global clock just keep on ticking and check all distributed application is synchronized after a tick. This is simple process so relative synchronization can be efficient in SIL simulation.

IV. CASE STUDY

The following cases are assuming distributed application in smart home environment.

A. Synchronization between two applications

Let's assume that there are two distributed applications. One is gas sensing application and the other is ventilator application. The control cycle of gas sensing application is 10ms and the control cycle of ventilator application is 30ms. Before simulation, the time of global clock and all local clock is 0ms as shown in Fig. 8. When simulation start, global clock start to tick continuously.

When global clock tick once, time of global clock passes as shown in Fig. 9. And then global clock send *tick* message to two distributed applications. The next time of gas sensing clock is equal to the time of global clock whereas the next time of ventilator clock is smaller than the time of global clock. So, as shown in Fig. 10, just gas sensing clock passes and its time is added its cycle, 10ms whereas ventilator clock keep on stopping. Though ventilator clock keep on stopping, there is no problem because this moment isn't time that ventilator application doesn't operate. When global clock ticks again, changes of clocks is the same as first tick of global clock as shown in Fig. 11. Ventilator clock still keep on stopping and clocks of other changes.

```
Initialize application;
Configure application;
Initialize local clock and set own control cycle;

**While** (waiting for *tick* message from global clock)
    **if** (time of global clock is equal to or greater than next time of local clock)
        Gathers data from other applications;
        Carry out own role;
        Tick local clock;
    **End if**
    Send *ticked* message from global clock;
**End While**

Destroy application;
```

Fig. 6. Pseudo code of synchronization algorithm in distributed application

```
Initialize global clock and set its cycle;

**While** (all distributed application is synchronized)
    Tick clock;
    Send *tick* message to all distributed application;
**End While**

Destroy global clock;
```

Fig. 7. Pseudo code of synchronization algorithm in global clock

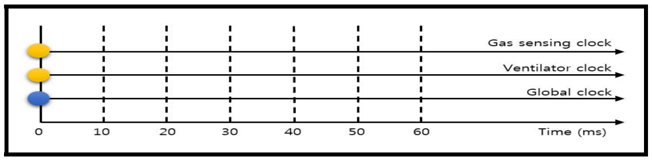

Fig. 8. Initial status of global clock and local clocks before simulation

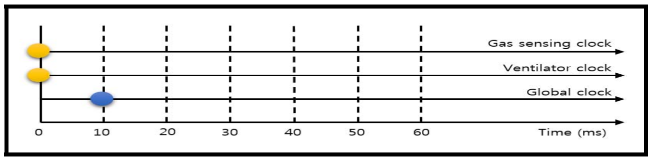

Fig. 9. Change of global clock after 1st tick of global clock

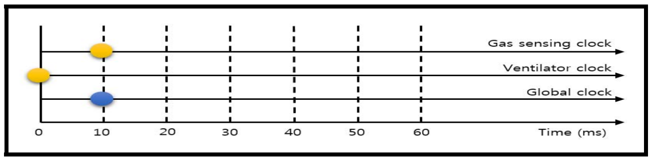

Fig. 10. Change of gas sensing clock after 1st tick of global clock

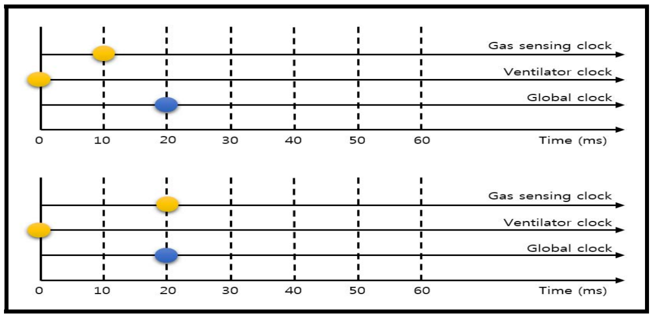

Fig. 11. Change of clocks after 2nd tick of global clock

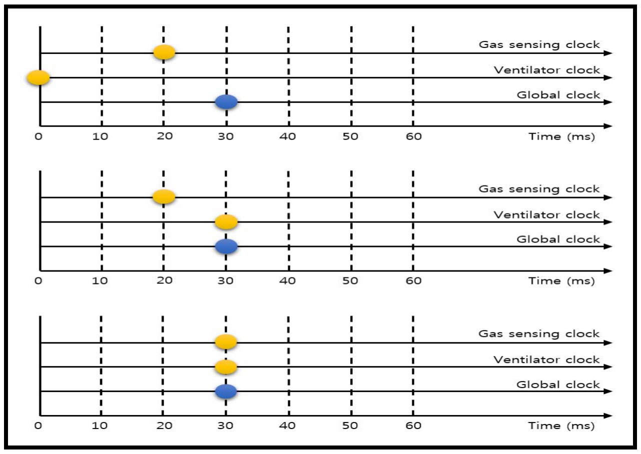

Fig. 12. Change of clocks after 3rd tick of global clock

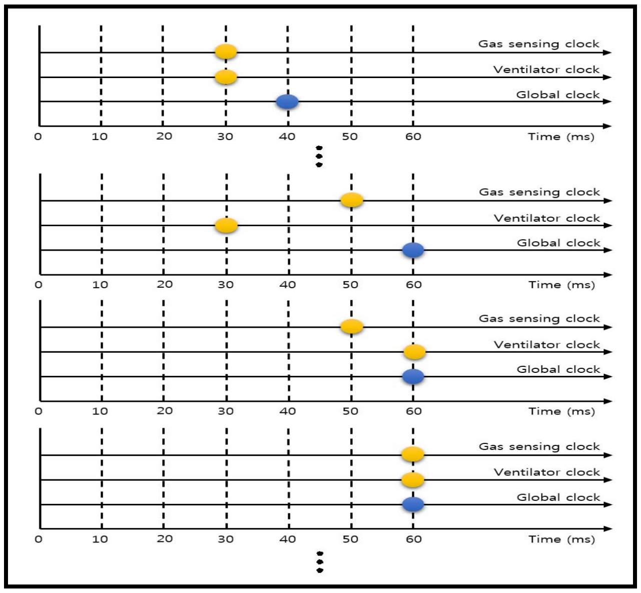

Fig. 13. Relative synchronization after 3rd tick

When third tick of global clock occurs, changes of clocks are different in previous ticks. When an application operates, with data from other applications which have gathered just before. In this case time of ventilator clock should pass before time of gas sensing clock passes. In real environment two applications concurrently operate. Ventilator application operate fans if gas sensing application detects gas. But at the moment, in the point of view of ventilator application, the detected data have gathered in the past. So ventilator clock passes and then gas sensing data passes as shown in Fig. 12. Since then, as shown in Fig. 13, relative synchronization continues.

B. Synchronization among many applications

Also synchronization among many applications occurs similarly. When global clock passes, among local clocks passed at that time, local clock of actuating applications should passes earlier than one of sensing applications.

V. Conclusion

In this paper, existing synchronization has high accuracy but require complicated calculation so it is inefficient for SIL simulation. Thus, we propose a method of relative time synchronization for SIL simulation. The method consider just relative sequence among distributed application for SIL simulation without complicated calculation. We showed that our method enables distributed applications to be synchronized. In the future, we study on distributed SIL simulator with synchronization method.

Acknowledgment

This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) (IITP-2016-H8601-16-1002) supervised by the IITP(Institute for Information & communications Technology Promotion) and Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(No. NRF-2014R1A1A2058733)

References

[1] E. Pak, Y. Ha, J. Park, Y. Kim, M. Song and T. Kim, “SYNDICATE: Software Platform for Distributed Real-Time System,” *Proc. 2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing(PRDC)*, pp. 327-328, Nov. 2015.

[2] J. Park and T. Kim, “A Method of Logically Time Synchronization for Safety-critical Distributed System,” *Proc. 2016 18th International Conference on Advanced Communication Technology (ICACT)*, pp. 356-359, Jau. 2016.

[3] C. Takahiro, I. Yuichi, M. Yoo and Y. Takanori, “A Distributed Real-Time Operating System with Location-Transparent System Calls for Task Management and Inter-task Synchronization,” *Proc. 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 1133-1138, Nov. 2011.

[4] Y. Pan, X. Yao, “Clock Synchronization in Distributed Real-Time Simulation System,” *Proc 2009 International Workshop on Education Technology and Computer Science(ETCS)*, Vol. 2, pp. 768-771, Mar. 2009.

[5] *IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pp. 31-36, Jul. 2013.

[6] G. Bam, E. Dilcan, B. Dogan, B. Dinc, B. Tavli, “DLWTS: Distributed Light Weight Time Synchronization for Wireless Sensor Networks,” *Proc. 2015 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pp. 447-450, Nov. 2015.

[7] C. Li, Y. Ofek, “Distributed Source-Destination Synchronization,” *Proc. 1996 IEEE International Conference on Communications*, Vol. 3, pp. 1341-1347, Jun. 1996.

[8] J. Kim, S. Seo, J. Chun, J. Jeon, “Distributed Clock Synchronization Algorithm for Industrial Networks,” Proc. *2010 IEEE International Workshop on Factory Communication Systems (WFCS)*, pp. 211-214, May. 2010.

[9] “1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,” IEEE STD 1588-2008, IEEE Instrumentation and Measurement Society, July. 2008.

[10] E. Y. Song, K. Lee, “An Application Framework for the IEEE 1588 Standard,” 2008 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, pp. 23-28, Sep. 2008.