

A Method of Logically Time Synchronization for Safety-critical Distributed System

Jeman Park*, and Taeho Kim*

*Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea

jeman@etri.re.kr, , taehokim@etri.re.kr

Abstract— Legacy real-time and embedded systems have been small scale and standalone. However, recently they are increasingly being connected to create large-scale distributed system. Such distributed embedded system is realized in automotive, medical, and manufacturing automation. Also, there are many researches about software platform for large-scale distributed system. Verification of the application is required in SW platform for safety-critical distributed system. However, the verification complexity increases in practice as several factors such as delay, jitter and error. An equivalent synchronized execution of distributed tasks are required in distributed real-time system. Our method is based on the PALS (Physically Asynchronous Logically Synchronous) which is a complexity-reducing architectural pattern for distributed real-time systems. It allows developers to design, verify and implement a logically synchronous implementation of real-time distributed applications. In this paper, we introduce a logical time synchronization for reducing complexity between distributed nodes in distributed embedded system. It make verification and development of distributed software easier. It is helpful to develop distributed embedded system which requires safety and reliability. This paper is useful for application programmer or system manager who wants to establish distributed embedded system which requires safety and reliability.

Keywords— Logical Time Synchronization, PALS, Middleware, Safe-Critical System, verification complexity

I. INTRODUCTION

In safe-critical system, verification of system stability is required for prevent an accident life and property. However, recently scale of systems are growing and distributing through networks [1], [2]. Such distributed embedded system is realized in automotive, avionics, medical, and manufacturing automation. For example, swarms of drones could fan out over an area to report the crimes, accidents, or weather pollution. Cars in the roads collaborate to keep the distance between cars thus preventing car accidents. For this reason, because of complexity of the system is increased [3]-[5]. It is difficult to development and verification. In this paper, we descript logically time synchronization of safety-critical distributed system. This leads to easier to verification of software, as a result, it is possible to ensure stability and reliability. It is useful for application programmer or system manager who wants to establish distributed embedded system which requires safety and reliability. In distributed system applied logically time synchronization, complexity of development

and verification of software decreases, thus ensuring safety through software verification.

The remainder of this paper is organized as follows. Section 2 reviews previous work about the PALS (Physically Asynchronous Logically Synchronous). Section 3 presents our logically time synchronization method. Finally this paper is concluded in Section 4

II. RELATE WORKS

The PALS system [6], [7] is applicable to hard real-time systems that guarantee real-time, reliable message transmission and bounded clock skews. It envisions a logically synchronous distributed real-time architecture, in which a group of periodic distributed tasks collaborate for fault-tolerant monitoring and control of physical systems and processes.

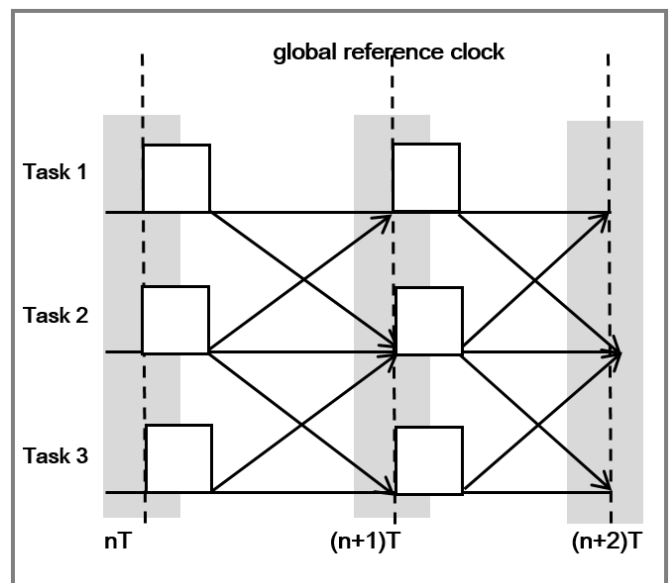


Figure 1. Ideal logical time synchronization

In the ideal logical time synchronization systems like a figure 1, the distributed tasks are triggered periodically by the global clock with a period T . Thus, these tasks send a communication messages at the same global time. At each messages, a task reads messages from its input ports, processes the messages and sends its output messages to other

nodes. In this lockstep execution, messages generated during the period n are always consumed by their destination tasks in the period $n + 1$.

The main concept of logical synchronization in the PALS system is simple and is illustrated in Figure 2. The PALS system guarantees an equivalent synchronous execution in the physically asynchronous architecture. In this architecture, the distributed tasks are now triggered periodically by the local clocks with the same period T . Since the local clocks are asynchronous, these tasks do not dispatch at the same global time. Despite these asynchronous dispatches, the PALS system will guarantee that these tasks dispatch within well-defined periodic intervals and messages generated during the period j are always consumed by their destination tasks in the period $n + 1$. As a result, the input views and operations of the tasks become identical to those in the synchronous model running with the same period. The PALS system must satisfy some timing and external input constraints to achieve the logical synchronization.

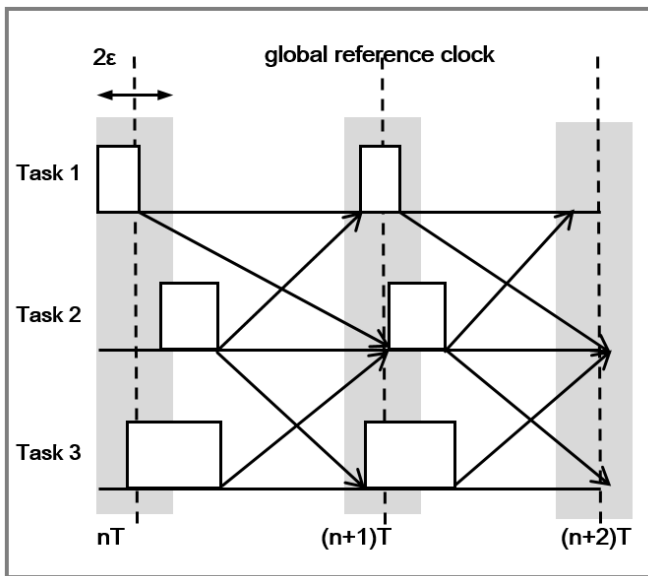


Figure 2. Logical time synchronization in real environment

III. LOGICAL TIME SYNCHRONIZATION BASED FOR REAL-TIME OPERATING SYSTEM

To guarantee time constraints of real-time tasks, Qplus-AIR [1] is integrated in the SYNDICATE platform by ETRI. It complies with ARINC 653 [8] standard which is a software specification for time and space partitioning in avionics. In this project, we extended Qplus-AIR as a distributed real-time operating system. Our method of logically time synchronization is based on SYNDICATE platform and PALS. It support a complexity-reducing architectural pattern that addresses the problem of physical asynchrony and guarantees consistency in real-time distributed computations. Engineers can use this pattern to develop distributed applications based on a globally synchronous system with a single global clock. The pattern executes the synchronous design on the physically asynchronous architecture by preserving the correctness. It

provides significant benefit in the system verification. The formal verification cost is greatly reduced since engineers verify only the simple synchronous model as long as the pattern's assumptions are satisfied. It coordinates the startup of the tasks participating in logically synchronous interactions. Since the a single global clock period T is an integer multiple of the task period $T(i)$, a task may start only at the next a single global clock event n after the startup. We define the local clock time of a global clock event, given by nT , as the global base time of the period n .

A. Framework of Logical Time Synchronization

In a RTOS (Real-Time Operating System), tasks are periodically executed by scheduler. Therefore, it is necessary an architecture different from general operating systems like a figure 3. A logical time synchronization based for RTOS is designed as three modules which are PALS integration module, communicating module, initial time synchronous module.

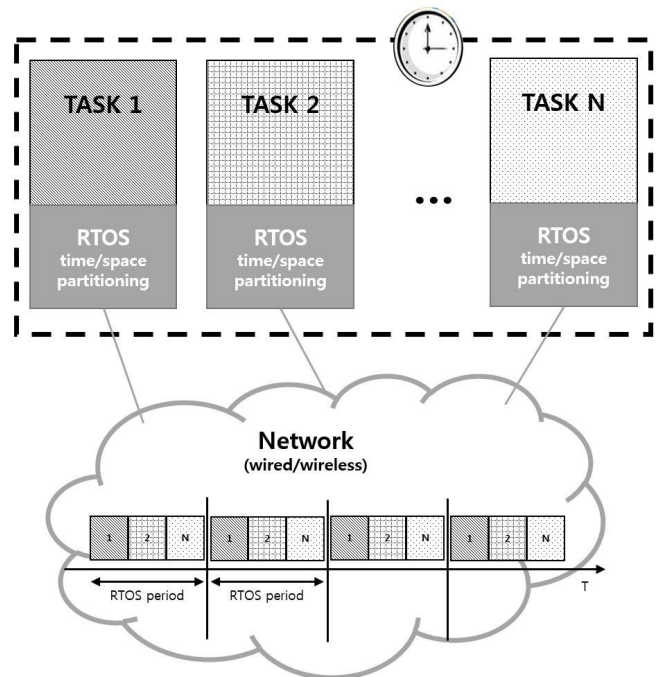


Figure 3. Framework of Logical Time Synchronization

RTOS in the SYNDICATAT platform has a MTF (Major Time Frame) that is defined as a multiple of the least common multiple of all partition periods in the module. Each major time frame contains identical partition scheduling windows. In PALS integration module, global master clock period is defined least common multiple of MTFs.

B. Period of RTOS Schedule and Global Time Clock

RTOS communication tasks are also executed periodically. There are two kind of tasks which are PALS task and Non-PALS task. Each system has own MTF and another booting time. Initial time synchronous module is align the start time of PALS and period of MTFs.

Figure 4 shows message communication flow in MTF. There is three kind of tasks which are PALS task, non-PALS task, and communication task. A non-PALS task does not need time synchronization. Each PALS and non-PALS task send messages to communication task via 'sampling API' or 'queuing API' based ARINC 653. Each communication task are connected with TCP and UDP.

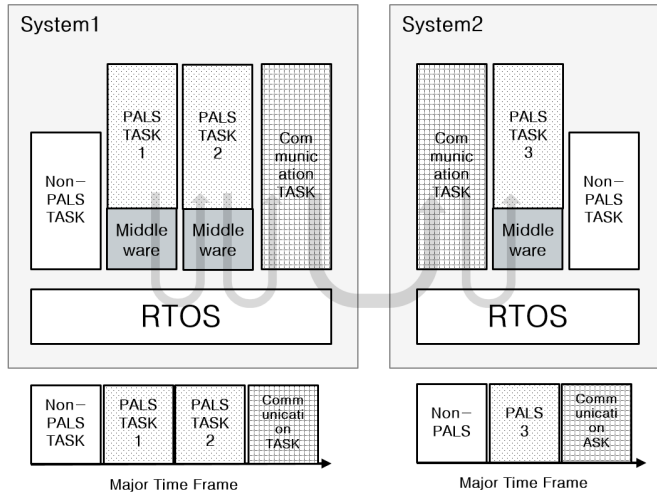


Figure 4. Message communication flow

Figure 5 shows an example for relation of MTF and global time clock. There are two systems and 3 PALS task. Each systems uses the local clock time (MTF) to execute the tasks. Each task is periodically executed as if it had started its first execution at the time of origin. Computation is triggered by a clock tick at each round. Computation changes the node's state and issues messages to other nodes. A message is destined at the next round.

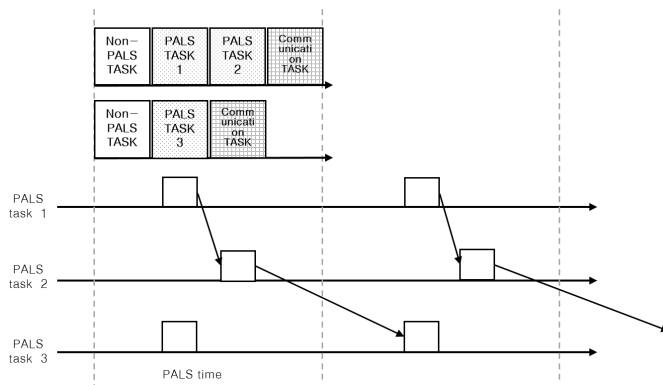


Figure 5. MTF and Global Time Clock

C. Communication Message Format

In order to exchange messages over a communication partition, common message format is needed. Figure 6 shows a communication message format for logically time synchronization.

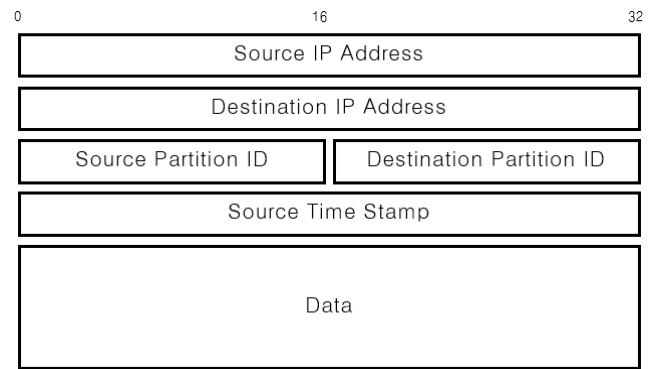


Figure 6. Communication message format

- Source IP Address: IP address of the system that belongs to a source task
- Destination IP Address: IP address of the system that belongs to a destination task
- Source Partition ID: Identification number of the partition that belongs to a source task
- Destination Partition ID: Identification number of the partition that belongs to a destination task
- Round Time: a number of round 'global time period'
- Major Time Frame: MTF period of system
- Source Time Stamp: a time of message generating
- DATA: messages

IV. CONCLUSION

Components in safety-critical distributed system require consistent views and actions in real-time to guarantee safety and correctness. In this paper, we design and implement logical synchronization of tasks for reducing complexity. For logical synchronization, it synchronizes the time which task releases its operation or tries communication with other tasks. When system starts, as multiple nodes complete its initialization and be ready at different time, arbitration for synchronous start is also required. As PALS model assumes underlying clock synchronization, middleware exploits advanced synchronization protocol. Our proposed solutions guarantee a logically synchronous abstraction for the real-time distributed computations. In this approach, we can reduce the amount of efforts spent for the distributed system design and verification. Engineers need to design and verify only the simple globally synchronous model as long as the system architecture satisfies pattern's assumptions.

ACKNOWLEDGMENT

This work was supported by IITP grant funded by the Korean government (MSIP) (No.B0101-15-0663, Safety-critical Distributed Modular SW Platform).

REFERENCES

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

- [2] T. Kim, D. Son, C. Shin, S. Park, D. Lim, H. Lee, B. Gim, and C. Lim, Qplus-AIR: A DO-178B Certifiable ARINC 653 RTOS, The 8th International Symposium on Embedded Technology (ISET) 2013
- [3] <https://www.tttech.com/products/product-filter/ttethernetproducts/>
- [4] <http://www.ethercat.org/default.htm>
- [5] http://en.wikipedia.org/wiki/Avionics_Full-Duplex_Switched_Ethernet
- [6] L. Sha, A. Al-Nayeeem, M. Sun, J. Meseguer, and P. Ölveczky. "PALS: Physically Asynchronous Logically Synchronous System," University of Illinois, Technical Report. May 2009.
- [7] A. Al-Nayeeem, C. Kim, W. Kang, P. Wu., "Middleware design for Physically-Asynchronous Logically-Synchronous(PALS) system," EMSOFT 2013, Oct. 4 2013
- [8] Airlines Electronic Engineering Committee, Avionics Application Software Standard Interface ARINC Specification 653 Part 1, Aeronautical Radio Inc., 2006



Jeman Park received his B.S., M.S. and PhD in Electronics Engineering from Hanyang University, Seoul, Korea, in 2014. Since 2012, he has been a member of embedded SW research department at ETRI, Daejeon, Korea. His current research interests include Future Internet, wireless sensor/ad hoc networks, 3G/4G cellular systems, Cyber Physical System, Real-Time Operating System.



Dr. Taeho Kim is a principal researcher and director of Next Generation OS research section at ETRI (Electronics and Telecommunications Research Institute), Korea. He graduated from KAIST (Korea Advanced Institute of Science and Technology) with a Ph.D in Computer Science at 2005. He was an international fellow at SRI (Stanford Research Institute) from 2001 to 2002, and he joined at ETRI at 2005. His research interests are dependable software infrastructure including embedded hypervisor, real-time operating systems, and formal verification.