Supplementary Material III: Case Study Note

2022. 7. 16.

Youngmok Ha

ETRI

History

Ver.	Date	내역	Author	
1.0	2022.07.16	Version 1.0	Y. Ha	

Contents

1. INTRODUCTION 4

1.1 PURPOSE 4

1.2 SCOPE 4

2. BACKGROUND 5

2.1 PHYSICAL SYNCHRONIZATION 5

2.2 LOGICAL SYNCHRONIZATION 5

3. EXPERIMENTAL SETUP 7

3.1 NETWORK IN THE REAL-WORLD 7

3.2 SYSTEM CONFIGURATION 9

3.3 EXPERIMENTAL TIME 10

3.4 COMPARISON GROUP 10

4. RESULTS 11

5. REFERENCES 13

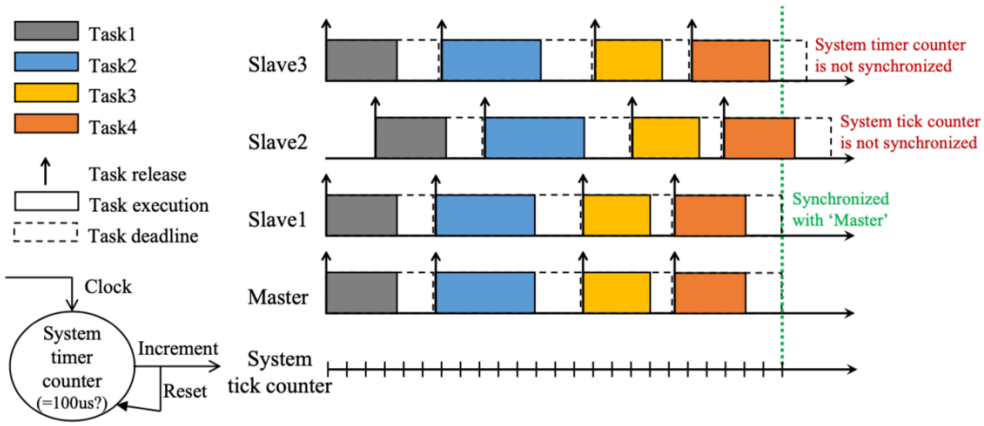
1. Introduction

1.1 Purpose

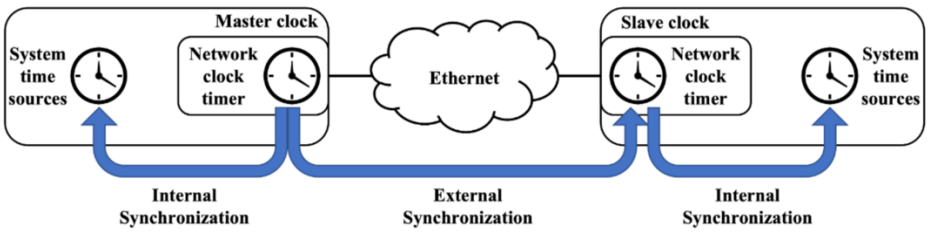
This document is written as a part of supplementary materials for the paper titled “Clock Offset Estimation for Systems with Asymmetric Packet Delays” submitted for publication in IEEE/ACM Transactions on Networking.

1.2 Scope

This document contains the detailed setup and results of the physical synchronization and logical synchronization among distributed real-time schedulers presented in Section VI of the paper. The physical synchronization includes an external (clock) synchronization over a real network with uncontrolled traffic and multiple hops between nodes, and an internal (clock) synchronization within the same node described in Figure 1. The logical synchronization implements the PALS (Physically Asynchronous Logically Synchronous systems) [52] adopted for avionic systems and cyber physical systems.



(a) Simple example of physically synchronized schedulers



(b) Simplified process of the physical synchronization

Figure 1 Physical synchronization between distributed real-time schedulers.

2. Background

2.1 Physical synchronization

The purpose of the physical synchronization among distributed real-time schedulers is to guarantee that the real-time scheduler of all nodes in a distributed system operates in synchronization by sharing a common notion of time in a master-slave structure as presented in Figure 1(a). Specifically, it attempts to ensure that time sources referenced by the real-time scheduler of each slave node, a system tick counter and a system clock timer counter, are synchronized with the master time sources. In the real-time scheduling of each node, the tick count value is directly used to determine the start and end times of the task execution. The tick count value is incremented by a timer interrupt that occurs when the value of the clock timer counter reaches a specified value, which is one hundred microseconds in our case by default. From the perspective of the distributed system, discrepancies between the tick count values can cause positive or negative delays in task scheduling. Offset and different speed of clock timer counters can induce different rates of the tick counting, longer or shorter task execution times, and postponed or advanced task deadlines.

The physical synchronization can be achieved by the external (clock) synchronization and the internal (clock) synchronization as depicted in Figure 1(b). The external synchronization is performed between the network clock timer counters of the master and slave nodes via Ethernet, and is achieved by clock synchronization methods including the proposed approach. The internal synchronization occurs between the network clock timer counter and system time sources within the same node, and is proceeded in the interrupt service routine (ISR) of the timer interrupt.

2.2 Logical synchronization

The PALS is a real-time logical synchrony protocol to support real-time global computation. Under the PALS protocol, engineers design and verify applications as if all the distributed state machines were driven by a single global clock. The PALS protocol is optimal in the sense that 1) the bound on the periods of the real-time global computation, such as the supervisory controller, is the shortest possible, and 2) the message overhead in achieving logical synchrony is minimal [52].

To help the reader understand the PALS without confusion, we summarize [52] and use the same symbols and notations as [52] only in this part. Note that we use the PALS clock period T defined by G7 as a performance indicator. If the reader is not interested in PALS, it is only necessary to familiarize with the simple symbols and notations in G5 and G7 to understand our paper.

[Summary of PALS]

A PALS system consists of state machines, environment input synchronizer, environment output synchronizer, and PALS clocks defined by rules G1, G2, G3, G4, G6 and G7:

G1: Local Clocks for Global Computation. Each state machine M_i engaged in global computation is driven by a local clock C_i with the same period T. The global clock time is denoted as t. Clock C_i is said to be at its j-th period, denoted as $C_i = j$, if the global time t satisfies the constraint, $\uparrow(C_i = j) \leq t < \uparrow(C_i = j + 1)$; where $\uparrow(C_i = j)$ is the rising edge of clock C_i, when it just enters its j-th period.

G2: Real-Time Network. The network has a network queueing (scheduling) delay q bounded by $0 < q_{min} \leq q \leq q_{max}$ and a network transmission delay μ bounded by $0 < \mu_{min} \leq \mu \leq \mu_{max}$.

G3: Real-Time Machine. Messages arriving at real-time machine M_i during M_i's j-th clock period are buffered. At $\uparrow(C_i = j + 1)$, M_i reads the messages from the buffer, carries out the computation, transitions to the next state, and sends output messages. The task completion time α, including real-time scheduling, computation, and I/O is bounded by $0 < \alpha_{min} \leq \alpha \leq \alpha_{max}$. Since state transitions are driven by the clock, a state machine M_i is at its j-th state, when its clock is at its j-th period.

G4: Environment Message I/O Synchronizer. Let the input synchronizer, $M_{I\text{-}syn}$, be a real-time machine as defined by G3. Messages from the environment are sent to the input synchronizer. Messages arriving at $M_{I\text{-}syn}$ during $\uparrow(C_{I\text{-}syn} = j) \leq t < \uparrow(C_{I\text{-}syn} = j + 1)$ are buffered. At $\uparrow(C_{I\text{-}syn} = j + 1)$, the input synchronizer reads buffered messages and forwards them to their destinations. When machines need to send messages to the external environment, they send them to the output synchronizer. Similarly, the output synchronizer, $M_{O\text{-}syn}$, reads messages at the rising edge of its clock tick and forwards them to the environment. The output synchronizer allows an external observer to have a synchronous view of the distributed states of a global computation.

G5: The Period of Perfectly Synchronized Clocks. Giving a set of perfectly synchronized clocks used to drive global computation, the clock period T should satisfy the constraint $T > \alpha_{max} + q_{max} + \mu_{max}$.

G6: PALS Causality Rule. A machine M_i at (PALS) clock period j cannot send a message earlier than $\uparrow(C_i = j) + H$, where $H = 2\varepsilon - \mu_{min}$.

G7: PALS Clock Period. PALS clock period. $T > 2\varepsilon + max(\alpha_{max} + q_{max}, H) + \mu_{max}$.

3. Experimental Setup

3.1 Network in the real-world

We used an open network shared by hundreds of computing nodes for research and development. Traffic over the network was real, not controlled. Multiple hops between the master and slave nodes were guaranteed by at least four networking devices, a router, an L3 switch, and two hubs, connected to the internet using different lines.

Figure 2 and Figure 3 respectively present one-way packet delay (OWPD) $D_{s,k}$ and $D_{d,k}$ of *sync* and *delay-req* messages generated by the IEEE 1588 Precision Time Protocol (PTP). Figure 4 describes the asymmetry $\lambda_k = D_{d,k} - D_{s,k}$. Figure 5 plots the PDF of absolute of the asymmetry $||\lambda_k||$.

The measurement period was 6 hours from 1 PM to 7 PM.

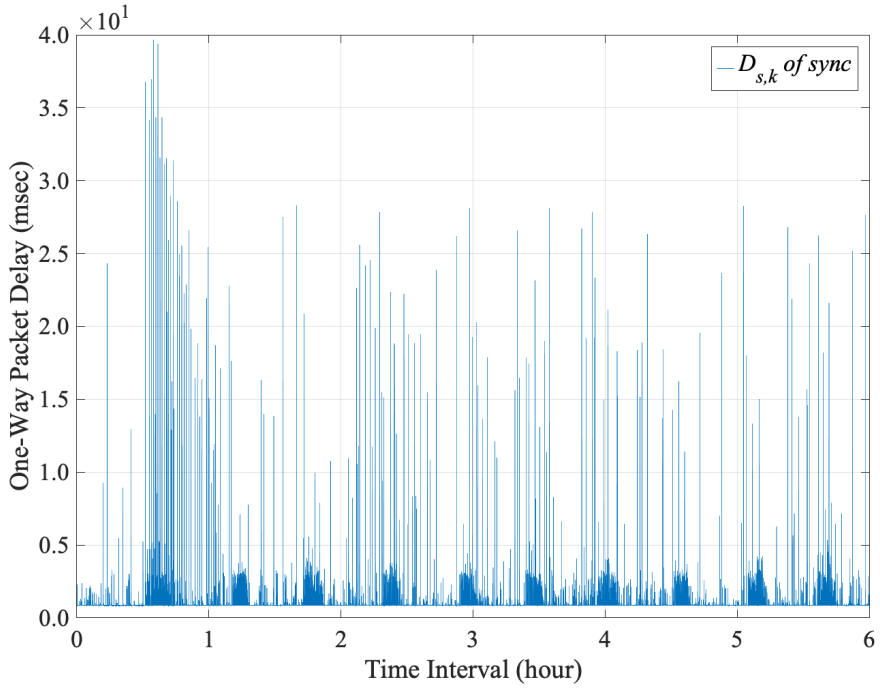


Figure 2. One-way packet delay $D_{s,k}$ of *sync* in the real network.

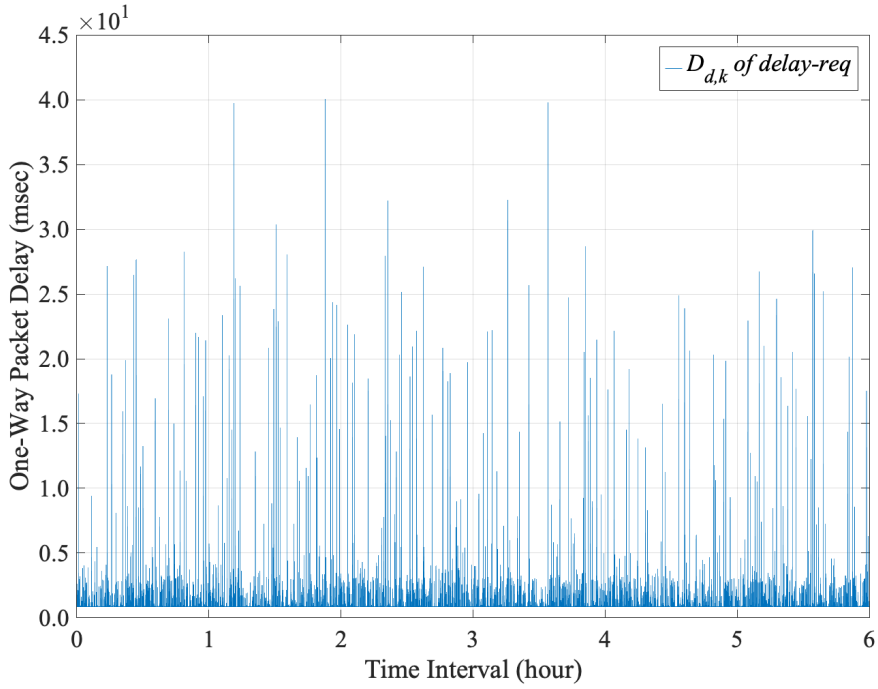


Figure 3. One-way packet delay $D_{d,k}$ of *delay-req* in the real network.

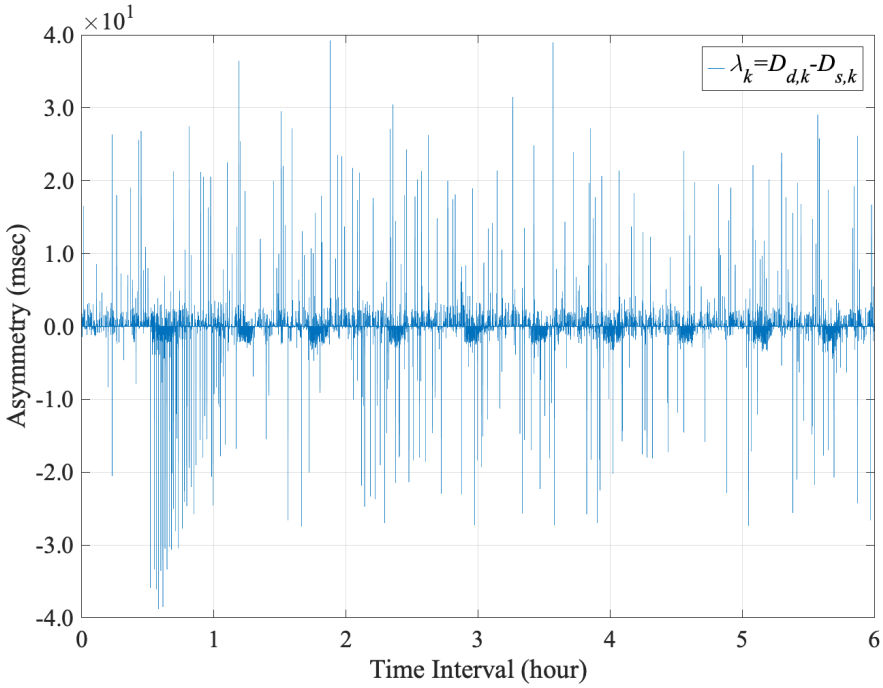


Figure 4. Asymmetry λ_k between $D_{s,k}$ and $D_{d,k}$ in the real network.

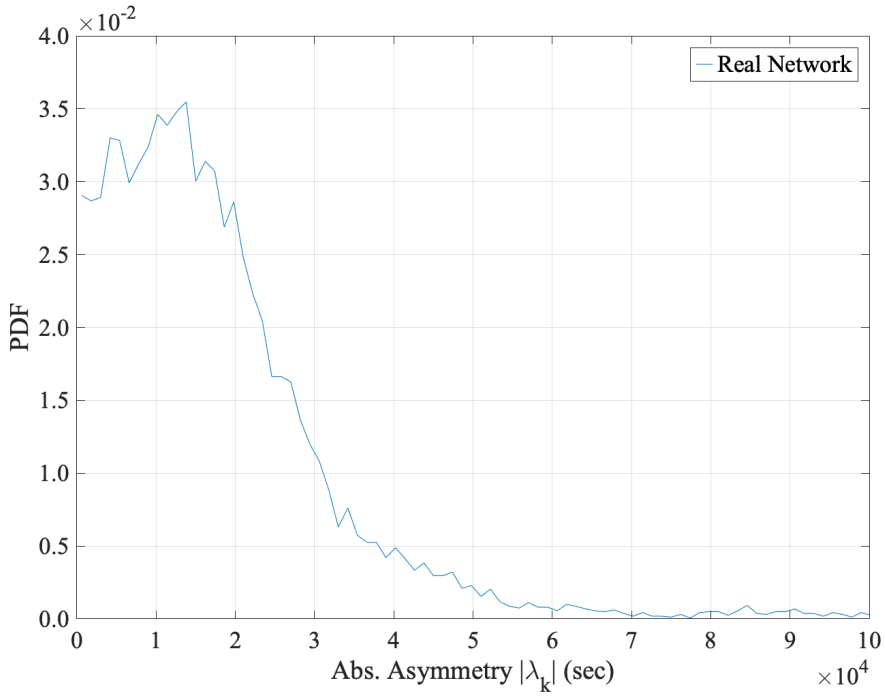


Figure 5. PDF of absolute of the asymmetry $|\lambda_k|$ in the real network.

Table I in this document provides the statistical profile of the real network environment. {Average, Minimum} of $D_{s,k}$ and $D_{d,k}$ were {1.18 × 10^{-3}, 8.24 × 10^{-4}} sec and {1.19 × 10^{-3}, 8.39 × 10^{-4}} sec, respectively. {Average, Minimum, Maximum}, and Variance of absolute of the asymmetry $||\lambda_k||$ were {5.95 × 10^{-4}, 1.86 × 10^{-9},3.92 × 10^{-2}} sec and 7.27 × 10^{-6} sec^2.

Metric	$D_{s,k}$	$D_{d,k}$	λ_k	$\|\lambda_k\|$
Average	1.18×10^{-3} sec	1.19×10^{-3} sec	1.29×10^{-5} sec	5.95×10^{-4} sec
Minimum	8.24×10^{-4} sec	8.39×10^{-4} sec	-3.89×10^{-2} sec	1.86×10^{-9} sec
Maximum	3.96×10^{-2} sec	4.01×10^{-2} sec	3.92×10^{-2} sec	3.92×10^{-2} sec
Variance	4.40×10^{-6} sec^2	3.72×10^{-6} sec^2	7.62×10^{-4} sec^2	7.27×10^{-6} sec^2

Table 1. The statistical profile of the real network environment

3.2 System configuration

We constructed a distributed real-time system composed of one master and three slave nodes connected by the real network. Four P2020RD-PCA boards acted as the master and slave nodes. On the boards, we installed a real-time operating system [53] developed by ETRI and commercialized by its spin-off company RTST. We ported the device driver 'gianfar' to the operating system for enabling the hardware timestamping and monitoring described in Section V-B of the paper. We also ported LwIP for networking and implemented a light-weight PTP stack for the experiments.

For the clock timer counters, we used ETSEC_1588_TMR_CNT_H/L and DEC, a decrementing counter of the e500v2 core, as the network clock timer counter and system clock timer counter, respectively. The system tick counter was incremented by one in the ISR of the decrementer interrupt in a scenario where the internal synchronization modifies the system tick counter and the decrementer auto-reload register (DECAR).

3.3 Experimental time

We run clock synchronization on the system for 6 hours during working time from 1 PM to 7 PM.

3.4 Comparison group

In addition to the proposed approach, previous approaches [1], [8], [10], [19], and [26] were implemented for performance comparison. Linux PTP [14] was not implementable because it was designed for Linux systems. PTPv2 [7] was also excluded because all the networking devices in the network must be PTPv2 compatible, and configuring the entire PTPv2 system is prohibitively expensive.

4. Results

Table 2 provides statistical results of the external synchronization in the real network environment. The results represent the performance of clock synchronization in a real-world network environment with uncontrolled traffic and multiple hops. In a real environment, the proposed approach demonstrated good performance. Our approach outperformed the other approaches. Although the distance between nodes and the experimental duration increased, its accuracy was maintained as that in Section V. For the metrics, the approach resulted in absolute errors of $7.54*10^{-8}$ sec, $1.77*10^{-15}$ sec^2, $2.06*10^{-7}$ sec. We analyze that the superior results were achievable because ζ_k dealt with the asymmetry of a total packet delay.

Metric	PTP [1]	Giorgi et al. [26]	Puttnies et al. [19]	Karthik et al. [8]	Proposed approach
Average	2.98×10^{-4} sec	7.76×10^{-3} sec	1.70×10^{-3} sec	2.30×10^{-5} sec	7.54×10^{-8} sec
Variance	1.82×10^{-6} sec^2	2.75×10^{-5} sec^2	1.85×10^{-6} sec^2	2.15×10^{-10} sec^2	1.77×10^{-15} sec^2
Maximum	1.96×10^{-2} sec	3.47×10^{-2} sec	1.08×10^{-2} sec	8.53×10^{-5} sec	2.06×10^{-7} sec

Table 2. The statistical results of external synchronization in the real network environment.

Table 3 provides statistical results of the physical synchronization among distributed real-time schedulers in the real network environment. For this, we used the following two metrics: the number of attempts required to modify the tick counter and the delays caused in task scheduling. Note that the modification can ruin task scheduling because the real-time scheduler selects the executed tasks by referencing the value. In the experiment, the proposed approach did not induce any modifications to the tick counter. This implies that our approach can set up environments wherein the schedulers perceive tick counters as linearly monotonic. During scheduling, the approach caused an average delay of $-1.97*10^{-8}$ sec. The result implies that our approach can provide time predictability for task scheduling on the order of tens of nanoseconds in a distributed system.

Metric	PTP [1]	Giorgi et al. [26]	Puttnies et al. [19]	Karthik et al. [8]	Proposed approach
Average # of Modifications	2134	16141	15561	0	0
Average Delays in Scheduling	1.49×10^{-5} sec	5.01×10^{-5} sec	4.91×10^{-5} sec	-2.30×10^{-5} sec	-1.97×10^{-8} sec

Table 3. The statistical results of physical synchronization between distributed real-time schedulers in the real network environment.

Table 4 provides statistical results of the logical synchronization in the real network environment. We used the period of the logical clock defined by G7 in [54] as a performance indicator. Generally, the length of the period is a factor affecting the computing efficiencies of distributed real-time systems. Similar to the low clock frequency of processing units, long periods of logical clocks could cause

performance degradation or even faults in distributed real-time computing. Compared with the ideal case of G5 presented in [54], the proposed approach lengthened the period by 3.91*10-7 sec from 4.46*10-2. From this result, we confirmed that we noted a performance penalty under 10-3% when the approach was applied to logical synchronization. Note that the period extension of 1.99*10-4 is the worst case because we used 100 us to set DECAR by default.

Metric	PTP [1]	Giorgi et al. [26]	Puttnies et al. [19]	Karthik et al. [8]	Proposed approach
Period Extension	1.99×10^{-4} sec	1.99×10^{-4} sec	1.99×10^{-4} sec	1.71×10^{-4} sec	3.91×10^{-7} sec
Performance Degradation	$4.48 \times 10^{1}\%$	$4.48 \times 10^{1}\%$	$4.48 \times 10^{1}\%$	$3.83 \times 10^{1}\%$	$< 10^{-3}\%$

Table 4. The statistical results of logical synchronization in the real network environment.

We conclude this document. The reasons why [26] and [19] performed worse than PTP have been provided on a Github page [52]. We do not deal with them in this document to avoid the repetition.

5. References

[1] “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,” IEEE Standard 1588–2002, Oct. 2002.

[7] “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,” IEEE Standard 1588–2008, Jul. 2008.

[8] A. K. Karthik and R. S. Blum, “Robust Clock Skew and Offset Estimation for IEEE 1588 in the Presence of Unexpected Deterministic Path Delay Asymmetries,” IEEE Transactions on Communications, vol. 68, no. 8, pp. 5102–5119, Aug. 2020

[10] S. Lee, S. Lee, and C. Hong, “An accuracy enhanced IEEE 1588 synchronization protocol for dynamically changing and asymmetric wireless links,” IEEE Communications Letters, vol. 16, no. 2, pp. 190–192, Feb. 2012.

[14] “The Linux PTP Project.” [Online]. Available: https://github.com/openil/linuxptp

[19] H. Puttnies, P. Danielis, and D. Timmermann, “PTP-LP: Using Linear Programming to Increase the Delay Robustness of IEEE 1588 PTP,” in 2018 IEEE Global Communications Conference. IEEE, 2018.

[26] G. Giorgi and G. Narduzzi, “Performance Analysis of Kalman-Filter Based Clock Synchronization in IEEE 1588 Networks,” IEEE Transactions on Instrumentation and Measurement, vol. 60, no. 8, pp. 2902–2909, Aug. 2011.

[52] L. Sha, A. Al-Nayeem, M. Sun, J. Meseguer, and P. C. Olveczky, “PALS: Physically Asynchronous Logically Synchronous Systems,” UIUC Technical Report, May 2009.

[53] E. Pak, D. Lim, Y. Ha, and T. Kim, “Shared Resource Partitioning in an RTOS,” in 13th Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT) in conjunction with the 29th ECRTS, 2017.