Implementing time-triggered communication over a standard ethernet switch

Kyriakakis, Eleftherios; Sparsø, Jens; Schoeberl, Martin

Published in:
Proceedings of the Fog-IoT Workshop 2019

Link to article, DOI:
10.1145/3313150.3313221

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

Implementing Time-Triggered Communication over a Standard Ethernet Switch

Eleftherios Kyriakakis
Technical University of Denmark (DTU)
Kogens Lyngby, Denmark
elky@dtu.dk

Jens Sparsø
Technical University of Denmark (DTU)
Kogens Lyngby, Denmark
jspa@dtu.dk

Martin Schoeberl
Technical University of Denmark (DTU)
Kogens Lyngby, Denmark
masca@dtu.dk

ABSTRACT

Cyber-physical systems in IIoT and Fog computing can use a variety of standards to guarantee real-time communication, based on enhanced switches with quality-of-service, audio-video bridging, time-triggered Ethernet and time-sensitive networking features. However, such real-time enabled network switches often come at a high design and maintenance cost.

This paper explores the feasibility of source time-triggered communication over a standard Ethernet switch without enhanced capabilities and demonstrates the minimum requirements of a system needed to enable time-triggered communication.

To achieve deterministic communication and synchronous operation of the tasks, the worst-case execution time of the tasks is analyzed and a static schedule is defined. We use the IEEE 1588 Precise Time Protocol to provide a global time reference for the network devices and time-triggered messages are scheduled at the source nodes. The communication is implemented and evaluated on a scalable synthetic cyber-physical system test-case composed of three nodes: a time server node and two application nodes that control a servo motor and exchange a time-triggered message containing the duty cycle of the pulse-width modulation signal.

CCS CONCEPTS

• **Computer systems organization** → **Embedded and cyber-physical systems**; **Real-time system architecture**; • **Networks** → **Network experimentation**; *Network protocols*.

KEYWORDS

Time-triggered communication, Cooperative tasks, Clock synchronization, WCET analysis, FPGA implementation

ACM Reference Format:
Eleftherios Kyriakakis, Jens Sparsø, and Martin Schoeberl. 2019. Implementing Time-Triggered Communication over a Standard Ethernet Switch. In *Workshop on Fog Computing and the IoT (IoT-Fog '19), April 15–18, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3313150.3313221

IoT-Fog '19, April 15–18, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6698-4/19/04…$15.00
https://doi.org/10.1145/3313150.3313221

1 INTRODUCTION

Emerging technologies in the field of industrial control and automation such as Fog computing and Industrial Internet-of-Things (IIoT) commonly incorporate or interface with distributed cyber-physical systems (CPS) that require deterministic machine-to-machine and machine-to-cloud communication [18, 22]. CPS networks are often composed of a set of sensor, actuation and monitoring devices that exchange time-critical information using some type of real-time communication protocol e.g., the time-triggered protocol TTP [7]. These real-time communication protocols support time-triggered (TT) communication based on a cooperative schedule and a network-wide synchronized notion of time [16].

TT communication has been investigated in a variety of protocols such as [2, 10], but in modern industrial and automotive networks, Ethernet is the preferred choice and is often provided by proprietary industrial Ethernet protocols such as PROFINET and EtherCAT [1]. Recently the increasing need for efficient systems interoperability and support for mixed-criticality traffic has given rise to two main advances in time-predictable Ethernet communication standards, the TTEthernet [6] and the IEEE 802.1 Time Sensitive Networking (TSN) task group [12].

TTEthernet was first introduced to allow for TT senders to operate over the same standard Ethernet links with event-based senders without interference and with guaranteed constant transmission delay and bounded jitter. TSN emerged from the automotive Ethernet application of the Audio-Video Bridging (AVB) and is in the process of developing a set of fully deterministic standards for local-area Ethernet networks that cover a range of topics from time synchronization using the IEEE 801.1ASrev standard to the IEEE 802.1Qbv, which allows for deterministic communication of time-triggered traffic. A close comparison of these two protocols is presented in [23].

Both standards describe the implementation aspects of end-system nodes and network switches as well as define non-functional requirements such as network routing, traffic flow, and topology. In [13] the authors investigated these non-functional real-time network requirements and described the necessary network planning to achieve bounded end-to-end communication in distributed CPS over TSN networks. In [19] the design of a custom TTEthernet switch is presented and compared against a commercial off-the-shelf switch. It is shown that the delays and jitter in the transmission time caused by a standard Ethernet switch's best-effort policy can lead to frames arriving outside of the receive-time window of a TT recipient.

Although TTEthernet and TSN have evolved as the main standards in the area of TT communication, their implementation comes at a high cost of design complexity and maintenance. This paper investigates the feasibility of time-triggered communication over

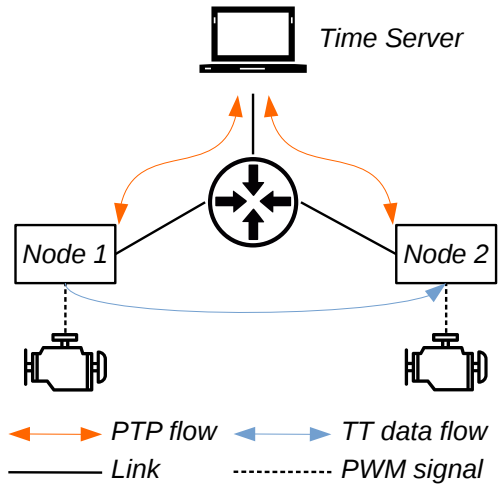

Figure 1: CPS test case network

a standard Ethernet switch without support for real-time networking. We explore the minimum requirements of a network composed of standard Ethernet switches and the design decisions needed for source scheduled TT communication, from an implementation point-of-view. We make use of hardware-based network time synchronization and worst-case execution time (WCET) analyzable software, and we measure the worst-case network propagation delay to create a static TT network and task schedule. To evaluate our approach we define and implement a test case of a distributed CPS application that allows us to check the correct transmission of TT messages and the synchronized operation of the tasks running on the network nodes.

2 THE TEST CASE

Figure 1 presents the network and communication flows of our test case CPS application. The network is composed of three network nodes, a time server that provides a global network time reference and two application nodes named, *Server* and *Client*.

Both of the application nodes drive a servo motor using pulse-width modulation (PWM). The PWM signal has a period of 20 ms and a duty cycle varying between 5% and 10% as shown in Figure 2. The duty cycle is increased by 1% on the *Server* and transmitted as a TT message to the *Client* with the goal of creating a synchronized rotation of the servos distributed on the two nodes. Both the *Server* and the *Client*, perform a PTP slave synchronization loop and also periodically report the application status on a HEX display. In summary, the two application nodes execute the following four tasks:

(1) `ctrl_task`, sends or receives the TT message containing the duty cycle
(2) `act_task`, controls the PWM signal of the servo motor
(3) `report_task`, reports the current clock offset from the PTP master
(4) `sync_task`, involves handling the time synchronization protocol

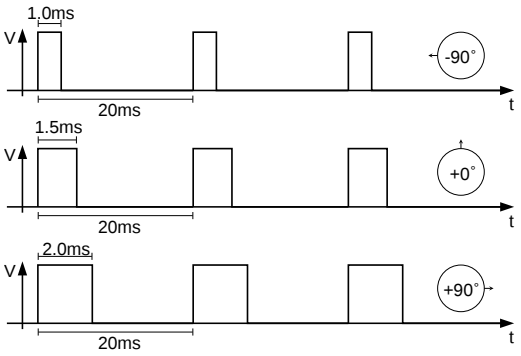

Figure 2: Servo motor control PWM signal

3 IMPLEMENTATION

This section presents the platform architecture of the nodes and describes the implementation of the application and the TT schedule.

3.1 Platform Architecture

The *Server* and *Client* nodes executing the tasks described in Section 2 are implemented on the open-source processor Patmos [15]. Patmos is a time-predictable, dual-issue, RISC processor that has been designed with focus on WCET analysis. Patmos uses special WCET-optimized instruction and data caches along with private scratchpad memories. An LLVM-based [9] toolchain, which includes the WCET analysis tool *platin* [4], supports Patmos.

As described in [16], it is necessary to achieve accurate time synchronization among cooperative network nodes before executing any distributed TT operations. We use the IEEE 1588v2 Precise Time Protocol (PTP) [11] to synchronize time over the network. The time server node acts as a time master while the application nodes act as PTP slaves.

To implement the TT communication over a general purpose Ethernet switch we make the following three assumptions: (1) all network devices have a common sense of time, (2) all network devices are cooperative senders and know the TT schedule and the resulting network activity, and (3) route paths between nodes are fixed with constant propagation delays.

3.2 Application Scheduling

The two application nodes initially execute a synchronization loop for a pre-specified number of clock cycles to synchronize their clocks to the global time reference provided by the PTP master that is configured on the time server node.

After the initial synchronization is complete, each application node proceeds to execute the static schedule defined in Figure 3 by polling the synchronized network time. The tasks have a period of 20 ms (constrained by the hard real-time requirement of the servo motor) and the schedule is composed of three sequential tasks, `ctrl_task`, `act_task`, and `report_task` followed by a period of clock synchronization tasks `sync_task`. This task sequence allows achieving the minimum clock offset between the nodes during the exchange of a TT message and a network synchronous generation of the PWM signals. In our implementation, the synchronized PTP

clock is used to generate the PWM signal in software as shown in Listing 1. This design choice allows for accurate synchronization of the distributed PWM signals in both the *Server* and *Client* nodes.

Listing 1: Software-based PWM signal using the IEEE 1588 clock

```
void exec_act_task(float dutyCycle){
    pwmTimer = get_ptp_usecs();
    *gpio_ptr = 0x1;
    while(get_ptp_usecs()-pwmTimer < HIGH_TIME(dutyCycle)
        -72){;}
    *gpio_ptr = 0x0;
}
```

To compensate for network propagation delays, the propagation delay of the network is measured during the initial PTP synchronization cycle by taking into consideration that a PTP packet has a maximum size of 54 bytes and that the transmitted TT message containing the duty cycle has a constant size of 46 bytes. We use the maximum delay observed to offset the schedule of the Client node. Moreover, the *Client* node's `ctrl_task` receive-time window is set to the WCET of the *Server* node's `ctrl_task`. Figure 3 (a) presents the network activity and the reserved PTP and TT slots. These time slots should be respected by other potential network users to avoid delaying any on-going transmission of TT messages that could lead to them being received by the *Client* outside the receive-time window. Figures 3 (b) & (c) present the respective task schedules for the *Server* and *Client* nodes.

Without a PTP-capable switch that can act as a boundary clock, PTP slave nodes synchronize their clocks from a single Ethernet interface on the Linux laptop that acts as a PTP master. As the PTP master broadcasts SYNC and FOLLOW_UP messages it receives DELAY_REQ messages from both nodes simultaneously to which it can only reply to one leading to un-replied messages from the PTP slaves. To mitigate this issue, during the synchronization task period, if a node gets successfully synchronized it stops and goes silent to allow for other network nodes to complete their PTP synchronization.

The WCET of the respective application nodes tasks is calculated using the tool *platin* [4] and is presented in the Table 1 and Table 2. The WCET presented for the `act_task` does not include the blocking wait time of the PWM signal (with a duration of `HIGH_TIME()`), i.e. it corresponds only to the clock cycles for executing the instructions.

Table 1: WCET Analysis of *Server* node periodic tasks

Task	WCET	
	Clock Cycles	Time (100 MHz)
`ctrl_task`	8396	83.96 μs
`act_task`	24752	247.52 μs
`report_task`	202	2.02 μs
`sync_task`	29066	29.525 μs

3.3 Source Access

The presented test-case application is implemented on the open-source project T-CREST [20] and the developed software application can be found at *https://github.com/t-crest/patmos/tree/master/c/apps/poorman-tte*. The PTP hardware-assist unit that was used for the time synchronization is hosted at *https://github.com/t-crest/patmos/tree/master/hardware/src/main/scala/ptp1588assist*

Table 2: WCET Analysis of *Client* node periodic tasks

Task	WCET	
	Clock Cycles	Time (100 MHz)
`ctrl_task`	8123	81.23 μs
`act_task`	24752	247.52 μs
`report_task`	202	2.02 μs
`sync_task`	29066	29.525 μs

4 EVALUATION

This section describes the experimental setup of our CPS demo network and presents the evaluation results of the clock synchronization and TT communication.

4.1 Experimental Setup

The presented CPS application is deployed on an experimental setup where the nodes are on the FPGA-based research platform T-CREST [14], which is implemented on two FPGA DE 2-115 Terasic boards equipped with an Altera Cyclone IV.

The time server (PTP master) is implemented on a Linux laptop, configured with the *linuxptp* package, that runs the *ptp4l* executable using the configuration presented in Listing 2.

Listing 2: PTP4l configuration file

```
[global]
verbose 1
delay_mechanism E2E
logAnnounceInterval 12
logSyncInterval -8
hybrid_e2e 1
twoStepFlag 1
time_stamping hardware
[enp0s31f6]
```

Both FPGA boards use a PLL to generate an internal system clock at a frequency of 100 MHz. Since the presented implementation makes use of FPGA technology combined with a standard Ethernet PHY, both nodes use a PTP hardware-assist unit [8] IP core to enable accurate clock synchronization. This IP core is similar in operation to commercial devices such as the Texas Instruments PHYTER [21].

The IEEE 1588-2008 clock of the PTP hardware-assisted unit is configured with a resolution of 20 ns. The network is communicating through an HP ProCurve 1700-8 [5] Ethernet switch at a bandwidth of 100 Mbps. The Ethernet MAC controller of T-CREST is configured with a single frame buffer leading to a tightly timed system, i.e., if a frame is not read by the processor in time and another Ethernet frame arrives it will be overwritten and missed.

4.2 Clock Synchronization

To evaluate the clock synchronization, of the two application nodes, the pulse-per-second (PPS) signals of the PTP hardware-assist units were connected to an oscilloscope that was set to trigger on the rising edge of the *Server* node's PPS. Figure 4 presents the comparison

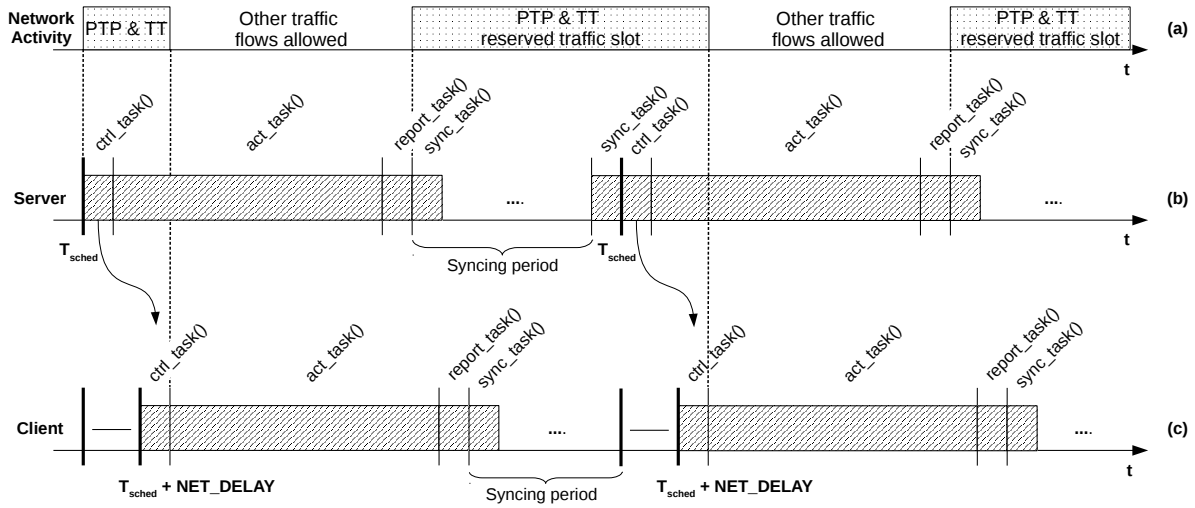

Figure 3: Client/Server node task scheduling and network activity

of the PPS signals produced by the respective *Server* (yellow) and *Client* (purple) *IEEE 1588-2008 hardware clocks* after the initial synchronization with the PTP master. The PPS relative offset is measured varying from −124.5 ns to 9.53 ns.

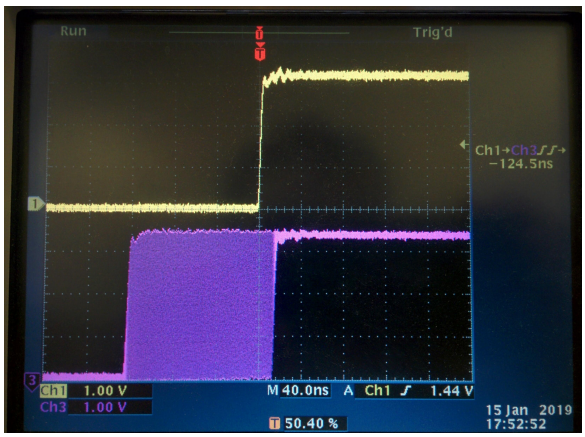

Figure 4: Comparison of PPS between PTP slave network nodes. Relative worst-case clock offset is measured at −124.5 ns

4.3 Time-Triggered Communication

The first results from the evaluation of the TT communication revealed that a receiving node should take into consideration for its schedule both the propagation delay of the switch and the WCET of the transmission task on the sending node to accurately tune its receive-time window. Otherwise, packets could arrive outside the expected time window of the receiver node and thus lead to missed frames. The worst case propagation delay for a single PTP packet of 54 bytes was measured at ≈23 μs. Furthermore, it was measured that non-cooperative nodes that transmitted during the `sync_task` duration greatly degraded the synchronization quality leading to either missed-packets on the receiver or miss-aligned PWM signal generation between the two nodes.

The presented time-triggered task schedule leads to a network activity that allows other cooperative senders to communicate during the silent time of the TT nodes, i.e. during the execution of internal tasks such as `act_task` and `report_task`. This closely resembles what is often described in real-time networking as a porosity schedule and its usage and design are discussed in [17].

5 CONCLUSION AND FUTURE WORK

This paper explored the engineering challenges of time-triggered communication and distributed task scheduling over a general purpose Ethernet switch. We implemented a synthetic CPS application and evaluated the design in an experimental setup composed of three nodes: a time server and two actuation nodes that control two servo motors. TT messages were exchanged from the server node to the client node controlling the duty cycle of two PWM signals. We defined a static schedule, we measured the worst-case network latency, we performed a formal WCET analysis and combined with the use of PTP and the *IEEE 1588-2008 hardware clock* to trigger the schedule execution allowed us to accurately synchronize the control of the two servo motors across our network and the time-triggered message communication.

Although we were able to evaluate the presented experimental TT communication scheme and gather first results on the negative effects of non-cooperative traffic on the TT flows, more work is needed in measuring the *Client* node's miss-rate and the achieved time synchronization [3] quality depending on the injected network traffic. Moreover, we plan to integrate the developed CPS test-case into a TSN network and based on the work presented in [13] we plan to investigate the design requirements of an end-system node that supports TT traffic. We plan to repeat the experiment within a TTEthernet network and conclude a set of software and hardware components for an end-system node needed to successfully achieve TT communication in real-time Ethernet networks.

ACKNOWLEDGMENTS

This research has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764785, FORA—Fog Computing for Robotics and Industrial Automation

REFERENCES

[1] Peter Danielis, Jan Skodzik, Vlado Altmann, Eike Bjoern Schweissguth, Frank Golatowski, Dirk Timmermann, and Joerg Schacht. 2014. Survey on real-time communication via ethernet in industrial automation environments. In *Proceedings of the Emerging Technology and Factory Automation (ETFA)*. IEEE, 1–8.

[2] Wilfried Elmenreich and Martin Delvai. 2002. Time-triggered communication with UARTs. In *Factory Communication Systems, 2002. 4th IEEE International Workshop on*. IEEE, IEEE, 97–104.

[3] Marina Gutiérrez, Wilfried Steiner, Radu Dobrin, and Sasikumar Punnekkat. 2017. Synchronization quality of IEEE 802.1 AS in large-scale industrial automation networks. In *Proceedings of the Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 273–282.

[4] Stefan Hepp, Benedikt Huber, Jens Knoop, Daniel Prokesch, and Peter P. Puschner. 2015. The platin Tool Kit - The T-CREST Approach for Compiler and WCET Integration. In *Proceedings 18th Kolloquium Programmiersprachen und Grundlagen der Programmierung, KPS 2015, Pörtschach, Austria, October 5-7, 2015*.

[5] Hewlett-Packard Development Company, L.P 2007. *ProCurve Series 1700 Switch Management and Configuration Guide*. Hewlett-Packard Development Company, L.P. Retrieved 14/01/2019 from ftp://ftp.hp.com/pub/networking/software/1700-MgmtCfg-Feb07-59916222.pdf

[6] Hermann Kopetz, Astrit Ademaj, Petr Grillinger, and Klaus Steinhammer. 2005. The time-triggered ethernet (TTE) design. In *Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*. IEEE, 22–33.

[7] H. Kopetz and G. Grünsteidl. 1993. TTP - A Time-Triggered Protocol for Fault-Tolerant Real-Time Systems. In *Proceedings of the 23rd Annual International Symposium on Fault-Tolerant Computing (FTCS '93)*, Jean-Claude Laprie (Ed.). IEEE Computer Society Press, Toulouse, France, 524–533.

[8] Eleftherios Kyriakakis, Jens Sparsø, and Martin Schoeberl. 2018. Hardware Assisted Clock Synchronization with the IEEE 1588-2008 Precision Time Protocol. In *Proceedings of the 26th International Conference on Real-Time Networks and Systems (RTNS '18)*. ACM, 51–60. https://doi.org/10.1145/3273905.3273920

[9] Chris Lattner and Vikram S. Adve. 2004. LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation. In *International Symposium on Code Generation and Optimization (CGO'04)*. IEEE Computer Society, 75–88.

[10] Roman Obermaisser. 2006. Reuse of CAN-based legacy applications in time-triggered architectures. *IEEE Transactions on Industrial Informatics* 2, 4 (2006), 255–268.

[11] Institute of Electrical and Electronics Engineers. 2008. *1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. IEEE.

[12] Institute of Electrical and Electronics Engineers. 2016. Time-Sensitive Networking Task Group. Retrieved 22/06/2018 from http://ieee802.org/1/pages/tsn.html

[13] Paul Pop, Michael Lander Raagaard, Silviu S Craciunas, and Wilfried Steiner. 2016. Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks. *IET Cyber-Physical Systems: Theory & Applications* 1, 1 (2016), 86–94.

[14] Martin Schoeberl, Sahar Abbaspour, Benny Akesson, Neil Audsley, Raffaele Capasso, Jamie Garside, Kees Goossens, Sven Goossens, Scott Hansen, Reinhold Heckmann, Stefan Hepp, Benedikt Huber, Alexander Jordan, Evangelia Kasapaki, Jens Knoop, Yonghui Li, Daniel Prokesch, Wolfgang Puffitsch, Peter Puschner, André Rocha, Cláudio Silva, Jens Sparsø, and Alessandro Tocchi. 2015. T-CREST: Time-predictable Multi-Core Architecture for Embedded Systems. *Journal of Systems Architecture* 61, 9 (2015), 449–471. https://doi.org/10.1016/j.sysarc.2015.04.002

[15] Martin Schoeberl, Wolfgang Puffitsch, Stefan Hepp, Benedikt Huber, and Daniel Prokesch. 2018. Patmos: A Time-predictable Microprocessor. *Real-Time Systems* 54(2) (Feb 2018), 389–423. https://doi.org/10.1007/s11241-018-9300-4

[16] Wilfried Steiner. 2007. Advancements in Dependable Time-Triggered Communication. In *Software Technologies for Embedded and Ubiquitous Systems*, Roman Obermaisser, Yunmook Nah, Peter Puschner, and Franz J. Rammig (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 57–66.

[17] Wilfried Steiner. 2011. Synthesis of static communication schedules for mixed-criticality systems. In *Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2011 14th IEEE International Symposium on*. IEEE, 11–18.

[18] Wilfried Steiner, Flavio Bonomi, and Hermann Kopetz. 2014. Towards synchronous deterministic channels for the internet of things. In *Proceedings of the IEEE world forum on Internet of Things (WF-IoT)*. IEEE, 433–436.

[19] Klaus Steinhammer, Petr Grillinger, Astrit Ademaj, and Hermann Kopetz. 2006. A Time-Triggered Ethernet (TTE) Switch. In *Proceedings of the Design Automation & Test in Europe Conference (DATE)*, Vol. 1. 1–6.

[20] T-CREST. 2017. Patmos Source. Retrieved 02/06/2018 from https://github.com/t-crest/patmos

[21] Texas Instruments 2015. *DP83640 Precision PHYTER–IEEE 1588 Precision Time Protocol Transceiver*. Texas Instruments. Retrieved 03/07/2018 from www.ti.com/lit/ds/symlink/dp83630.pdf

[22] TTTech. 2015. Deterministic Ethernet & TSN: Automotive and Industrial IoT. *Industrial Ethernet Book* 89 (July 2015). Retrieved 03/07/2018 from https://www.tttech.com/fileadmin/content/general/secure/pdf/IEB89_2015-Deterministic-Ethernet-TSN_Automotive-and-Industrial-IoT.pdf

[23] Lin Zhao, Feng He, Ershuai Li, and Jun Lu. 2018. Comparison of Time Sensitive Networking (TSN) and TTEthernet. In *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*. IEEE, IEEE, 1–7.