Clock Synchronization in Integrated TSN-EtherCAT Networks

Daniel Bujosa Mateu*, Daniel Hallmans*†, Mohammad Ashjaei*
Alessandro V. Papadopoulos*, Julian Proenza‡, Thomas Nolte*
*Mälardalen University, Västerås, Sweden
† ABB AB, Ludvika, Sweden
‡ University of the Balearic Islands, Palma, Spain

***Abstract*—Moving towards new technologies, such as Time Sensitive Networking (TSN), in industries should be gradual with a proper integration process instead of replacing the existing ones to make it beneficial in terms of cost and performance. Within this context, this paper identifies the challenges of integrating a legacy EtherCAT network, as a commonly used technology in the automation domain, into a TSN network. We show that clock synchronization plays an essential role when it comes to EtherCAT-TSN network integration with important requirements. We propose a clock synchronization mechanism based on the TSN standards to obtain a precise synchronization among EtherCAT nodes, resulting to an efficient data transmission. Based on a formal verification framework using UPPAAL tool we show that the integrated EtherCAT-TSN network with the proposed clock synchronization mechanism achieves at least 3 times higher synchronization precision compared to not using any synchronization.**

***Index Terms*—Time-sensitive networking, TSN, clock synchronization, EtherCAT, formal verification.**

I. INTRODUCTION

Technology is in continuous development and when new technical know-hows become available for use, this often opens up possibilities to design a new types of solutions. For manufacturers of systems or products, the availability of new technical solution provides a possibility to get an advantage over competitors. However, the opportunities of new solutions always come with the challenge of their integration with existing legacy solutions and implementation. Such challenges are common in large industrial systems and solutions where everything is not (or cannot be) changed at once, when a new technology is going to be utilized.

One of the new technologies, which offers a set of efficacious features for industrial systems, is Time-Sensitive Networking (TSN). It was introduced by the IEEE TSN task group[1] in 2012, presenting several interesting features such as offline scheduled traffic and support for frame preemption. TSN seems promising to enable new solutions within the context of modern industrial systems and solutions. TSN allows for multiple flows of time critical traffic, subject to requirements on bounded latency, to share the same network as generic traffic. Such capabilities are hinting towards the possibility to integrate multiple legacy networks onto one TSN network. However, it is common that legacy technology currently in use does not support all TSN requirements. Moreover, for companies it is cost-effective and beneficial if they gradually move towards new technologies instead of fully replacing the existing ones. Therefore, solutions towards integrating a legacy systems onto a TSN network in a way that the services are not disturbed are essential.

One of the vastly used industrial communication technologies in industrial systems, in particular in the automation domain, is EtherCAT (Ethernet for Control Automation Technology)[2] [1]. EtherCAT was introduced in the market in 2003, and its main advantage is the short latency that is imposed to the message frames due to the on-the-fly read and write procedure. In short, a train of frames, known as a *telegram*, is initiated by a master node and circulated in the network. The telegram passes through all slave nodes and, while passing through a slave node, the data can be read, written and updated with a latency that is only posed by the hardware propagation delay. Another feature given by the EtherCAT technology is that it provides clock synchronization among the slave nodes and the master node. Such clock synchronization is commonly used in the system where EtherCAT is deployed and hence must also be supported in the case of adopting a potential replacing technology such as TSN.

Contributions. In order to allow industry to adopt TSN solutions, a proper integration methodology should be designed. In this paper, we consider EtherCAT as a legacy network in an automation industry. Among different requirements in integrating EtherCAT devices onto a TSN network, an essential component is the clock synchronization to maintain proper behavior of the communication among the EtherCAT devices. Thus, the main target of this paper is the clock synchronization requirements for such integration. The main contributions of this paper are as follows:

- We formulate the problem of having inconsistent clock synchronization mechanisms in an integrated EtherCAT-TSN network and we describe the effects of this inconsistency in the network behavior.
- We propose a solution to integrate the clock synchronization mechanisms described by the two network technologies, i.e., EtherCAT and TSN, to obtain a precise synchronization.

[1]https://1.ieee802.org/tsn/

[2]https://www.ethercat.org/en/technology.html

- We model three different architectures including: (i) a solely EtherCAT network, (ii) an integrated EtherCAT-TSN network without clock synchronization mechanism, and (iii) an integrated EtherCAT-TSN network with our proposed clock synchronization solution. The modeling is based on a formal verification framework, using UPPAAL[3], to show the performance of the network with respect to the clock precision.

Outline. The paper is organized as follows. Section II describes the basics of clock synchronization in both TSN and EtherCAT. Section III presents the related work. Section IV formulates the problem, while Section V proposes our solution. Then, utilizing a formal modeling framework, Section VI evaluates the solution. Finally, Section VII concludes the paper and gives future directions.

II. Basics of Clock Synchronization Protocols

This section presents the background information on clock synchronization for both technologies highlighted in this paper, i.e., EtherCAT and TSN. The information gives basis for the proposed solution in this paper.

A. EtherCAT Clock Synchronization

According to its specification, EtherCAT presents three different synchronization modes: (a) free run, (b) synchronous with Synchronous Message (SM) event, and (c) synchronous with Distributed Clock (DC) SYNC event.

a) Free run mode: In free run mode there is no synchronization between nodes in the network. In this mode, all clocks in the nodes run independently, hence there are no timing-related properties provided. The other two modes provide different levels of synchronization, which will be described in more details below.

b) Synchronous with SM event mode: In this mode the slave nodes are synchronized with the master node by means of SMs that are sent by the master node. The SM messages are used for two different purposes. The first purpose is to exchange data among the slave nodes and the master node, whereas the second purpose is to use the same messages for synchronization among the slave nodes and the master node.

The main drawback of this mode is its low precision which is in the level of a few microseconds. The low precision in this mode is the consequence of the high level of jitter for the SM messages. The main reason is that typically the EtherCAT master nodes are implemented in standard PCs with network interfaces that do not support low-jitter communication.

c) Synchronous with DC SYNC event mode: In this mode a dedicated message is used for clock synchronization. The main advantage of this mode is its high precision compared to the SM event mode, which is in the level of nanoseconds whereas the SM event mode provides a level of a few microseconds. To achieve this precision, the master node executes the Delay Measurement (DM) mechanism, shown in Fig. 1.

The master node measures the delays between the reference clock (notated by RC in the figure) and the slave clocks (notated by SC in the figure). The reference clock is typically the clock of the first slave node in the EtherCAT strand, while the slave clocks are the clocks in the other slave nodes of the EtherCAT strand. This mechanism initiated by the master node sends a special message, known as the `DM Transmission` (DM_T). When the slave nodes supporting the DC SYNC receive this message, they record the time ($T1$ or $T2_i$ depending on whether the slave node has the reference clock or not). Once the message arrives to the last slave node, the message has to return to the master node. At this moment it is renamed to `DM Response` (DM_R). When DM_R message is received by a slave node, the slave node records the time in which the message was received ($T4$ or $T3_i$ depending on whether the slave node is the reference clock or not). With these values, the local delay of the RC (LDRC) and the local delay of the SC (LDSC), the delay between the reference clock and the slave clocks is calculated by the EtherCAT master node by: $\frac{T4-T1-T3_i+T2_i-LDRC-LDSC_i}{2}$.

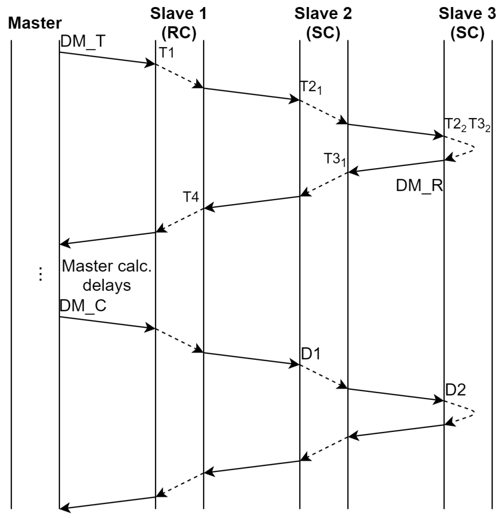


Fig. 1: Delay Measurement Mechanism.

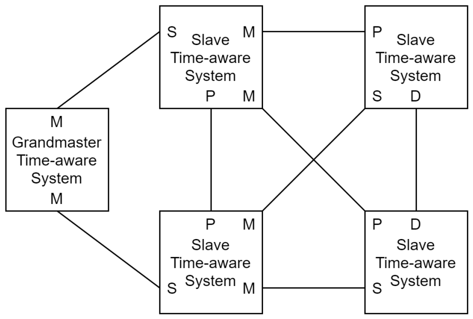
The master node transmits this value to the slave nodes by means of another dedicated message, known as `DM Calculated` (DM_C). At this point the master node periodically sends a SM message. The period of this transmission depends on the precision that is desired. This message records the local time of the reference clock by its reception. Then, when the message arrives at the other slave nodes, they add their delays to the local time of the reference clock saved in the message, and they update their internal clock. Finally, note that the EtherCAT master node can be synchronized with slave nodes by becoming a DC slave.

B. TSN Clock Synchronization

The mechanism providing the TSN clock synchronization (gPTP) is described in the IEEE 802.1AS standard and consists of three main parts including the Best Master Clock Algorithm (BMCA), the Propagation Delay Measurement (PDM) mechanism and the Transport of Time-synchronization Information

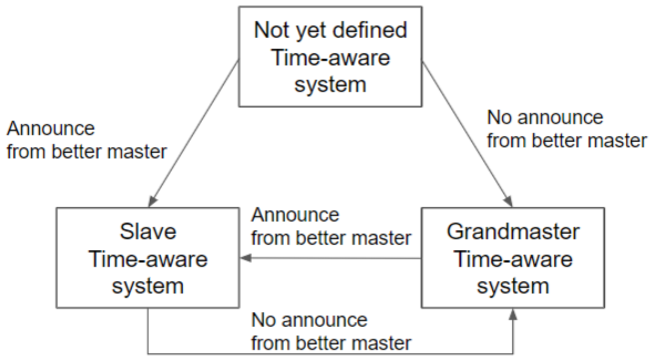
[3]http://www.uppaal.org/

(TTI). BMCA is used to determine the grandmaster clock, which is the reference clock in the TSN network, as well as the hierarchy between the different time-aware systems. Time-aware systems are the nodes in a TSN network that supports clock synchronization and scheduled traffic transmission. The PDM mechanism is used once the hierarchy is established in order to measure the propagation delay between systems. The TTI mechanism is used to forward the grandmaster time to synchronize the others time-aware systems. All three mechanisms are presented in more detail below.


S Slave Time-aware System M
P M
P Slave Time-aware System
S D
M Grandmaster Time-aware System M
P M Slave Time-aware System S M
P D Slave Time-aware System S
Fig. 2: Example of TSN time-synchronization spanning tree.

1) BMCA: The BMCA constructs a time-synchronization spanning tree with the grandmaster as the root. One example of this can be seen in Fig. 2. In the figure, we can note two remarks. First, each system can be either grandmaster time-aware systems or slave time-aware systems. The second observation is that the system ports can be Master ports (M), Slave ports (S), Passive ports (P) or Disabled ports (D). To determine all these behaviors, each system sends a special broadcast message called `announce message` periodically. The `announce message` contains different parameters and in this paper, for the sake of simplicity, we focus only on two of them: `systemIdentity` and `stepsRemoved`. `systemIdentity`, specifies how good the clock of the system sender of the message is, while `stepsRemoved` indicates how far the receiver is from the transmitter. Specifically, `stepsRemoved` is increased every time the `announce message` is forwarded. This means that if we have a line topology with three systems, and the first system in the line transmits its `announce message`, the message arrives to the second system with a value in the parameter `stepsRemoved` equal to 0. However, this system will increase `stepsRemoved` before forwarding it to the next system. Thus, the last system is going to receive the `announce message`, sent by the first system, with a value in the parameter `stepsRemoved` equal to 1.

As it can be seen in Fig. 3, when a system does not have an assigned role, it can become a slave time-aware system if it receives an `announce message` from a better clock, which means a better `systemIdentity` parameter, or it can become a grandmaster if, after a defined period of time (defined by the periodicity of which the `announce message` is transmitted), it does not receive any `announce message` from a better clock. If a system is a grandmaster or a slave time-aware system, then something similar can happen. If a granmaster time-aware system receives an `announce message` from a better clock, it becomes a slave time-aware system. In case a slave does not receive an `announce message` from a better clock after a defined period of time, it becomes a grandmaster time-aware system. On the other hand, depending on the proximity to the grandmaster, ports in a system have different roles and this proximity can be determined thanks to the `stepsRemoved` parameter. Specifically, the port closest to the grandmaster clock becomes the slave port, and only one port in the system can present this role. In a link, the port closest to the grandmaster clock becomes a master port. Finally, if a port is disabled, it becomes a disabled port and if it is none of the master, slave or disabled ports, then it becomes a passive port.


Not yet defined Time-aware system
Announce from better master
No announce from better master
Slave Time-aware system
Announce from better master
Grandmaster Time-aware system
No announce from better master
Fig. 3: Time-aware system BMCA evolution.

2) PDM: Once the spanning tree with the grandmaster as the root is created, the slave time-aware systems can carry out PDM to measure the propagation delay. This process is shown in Fig. 4. PDM starts with one system sending a delay request `Pdelay_request` through its slave port to another system, which can be the grandmaster or another slave time-aware system, and records the time when the message was transmitted ($T1$). The responder receives the message through its master port, records the time when the message was received ($T2$), sends $T2$ back to the initiator and records the time when the message was transmitted. The initiator receives $T2$ and records the time when the message was received ($T4$). Finally, the responder sends $T3$ to the initiator, so it calculates the delay by $Delay = \frac{(T4-T1)-(T3-T2)}{2}$.

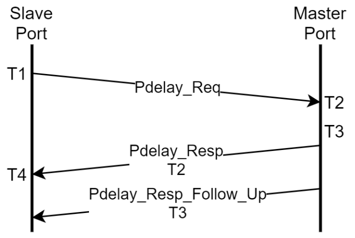


Fig. 4: PDM diagram.

3) TTI: Once the spanning tree with the grandmaster as the root is created and the slave time-aware systems have measured the delays, TTI is executed. TTI consists of systems sending their local time through their corresponding master ports to the systems connected to them. The systems that receive the message through their slave port adds the delay measured and update their local time accordingly.

III. Related Work

The TSN task group[4] was formed in 2012 with the aim of extending industrial network standards with support of time sensitive traffic, e.g. time-triggered transmission on top of the other traffic classes, scheduled traffic, frame preemption support and clock synchronization. There is a lot of research ongoing around TSN features, e.g. studying the effects of time-aware shapers [2], fault tolerance issues [3], scheduling policies [4] and load balancing in TSN networks [5]. The EtherCAT foundation has been part of TSN standardization and working on EtherCAT TSN Communication Profile, ETG.1700 S(D) V0.9.1[5]. One of the main challenges is to add stream adaptation logic in the network to be able to translate EtherCAT frames to TSN frames and vice versa. The proposed solution by the EtherCAT foundation describes a segment identifier to be added in the destination MAC addresses as the information is not changed during the transmission of the frames. Then, in each stream adapter the TSN VLAN tag will be added or removed. A special device is developed, e.g., EK1000 by Bechoff AG, to serve the stream adaptation.

One of the key features with EtherCAT is the clock synchronization between master and slave nodes or only among slave nodes. Most of the works in the literature are focused on studying and improving the performance of the EtherCAT networks. For example, the work in [6] evaluates the EtherCAT synchronization mechanism based on experiments, while the work in [7] shows the effects of using different synchronization schemes on the end-to-end traffic latency. The effects of using distributed clocks are also studied in [8]–[10]. Several works addressed the problem of improving clock synchronization in EtherCAT devices from different point of view. In this context, the work presented in [11] actively measures the synchronization error and compensates the error with a proposed mechanism. Moreover, the work presented in [12] proposed to use a central oscillator to coordinate the clocks among the nodes. The work in [13] proposed a method to integrate the clock synchronization with control loops to improve the precision of the synchronization. Most of the above mentioned works evaluated their proposals based on simulation experiments.

A major problem in the proposed synchronization methods and improvements is the fact that in most of the cases the EterCAT master node is implemented on a general-purpose hardware with a real-time operating system, e.g., RT-Linux, hence no precise clock can be obtained. In these cases, a jitter can be created up to $18\mu s$ dependening on the hardware and the master node configurations, according to [14]–[17]. By improving the master node, e.g., by replacing it with a special-purpose hardware, the precision of $20ns$ can be achieved [18].

[4]https://1.ieee802.org/tsn/

[5]https://www.ethercat.org

According to our survey, and to the best of our knowledge, there is no work addressing the challenges of integrating EtherCAT devices onto a TSN network, in particular from the clock synchronization perspective, except the frame adaptation identified by EtherCAT TSN Communication profile.

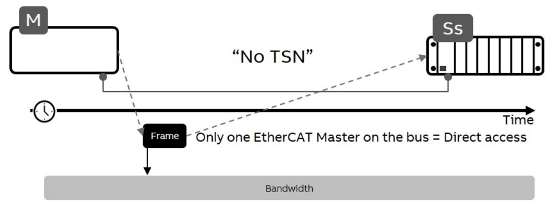
IV. Problem Description

Considering the network technologies in the types of systems that we target, if a legacy installation that is built around the EtherCAT technology is updated with a TSN network, or if a new system is designed that combines both TSN and EtherCAT networks, two key challenges must be considered, including stream adaption and clock synchronization. The former challenge is necessary to be addressed, e.g. by adding EK1000 hardware[6], or by providing adaption directly in the TSN bridge port connected to the EtherCAT devices, such that the TSN network will be able to handle the EtherCAT telegrams, as pointed out in Section III. The latter challenge concerning clock synchronization requires solutions to handle new sources of jitters that are introduced by the Time-Division Multiplexed (TDM) behavior of the TSN network.

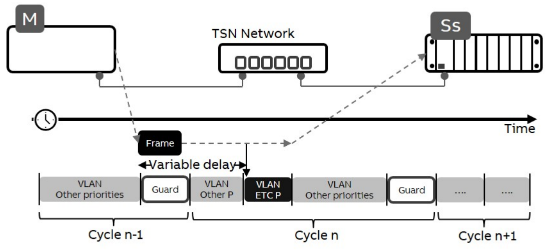
Fig. 5 shows three scenarios of connecting a master node (M) to multiple slave nodes (Ss) in an EtherCAT network. In Fig. 5a, the master node is directly connected to the slave nodes, while in Figs. 5b and 5c the EtherCAT nodes are connected through a TSN network. The difference between the two latter scenarios is if clock synchronization is present (Fig. 5c) or not (Fig. 5b). Additionally, in each scenario we can see how the bandwidth is managed and utilized in the network. In case of no TSN network integrated into the EtherCAT network, the master node transmits EtherCAT frames through the network, utilizing any available bandwidth in the network without any interruption, consequently resulting in very low jitter for the frames. The low amount of jitter in the EtherCAT network is because of the hardware interface of the EtherCAT master node. However, when the TSN network is integrated into the EtherCAT network the frame transmissions are coordinated by time slots that are configured by the TSN network. Note that a TSN network defines gate mechanisms resulting to time slots reservation for frame transmission. Therefore, a generated frame by a master node within the EtherCAT network may experience variation of delays, known as jitters, as shown in Fig. 5b.

The jitters that are added to a closed control loop can, in the worst case, result in a control system that is unstable and thereby unable to provide the designed efficiency. More details in this regard will be described in Section V. However, even if the introduced jitter caused by lack of clock synchronization can be ultimately tolerated by the control loops in a particular application, the control loop stability analysis (conducted at design time) must be re-assessed, which can result in a situation where the entire system must be verified from scratch, causing a significant investment cost. This cost is likely to be higher in the case of large legacy systems.

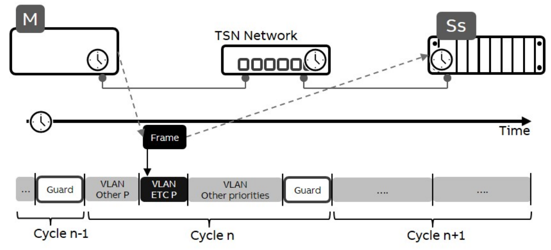
[6]https://www.beckhoff.com



(a) EtherCAT network behavior without TSN network.



(b) EtherCAT network behavior with TSN network but without clock synchronization between them.



(c) EtherCAT network behavior with TSN network and clock synchronization between them.

Fig. 5: EtherCAT network behaviors depending on the presence of a TSN network, possibly with clock synchronization.

Due to the above-mentioned reasons, it is essential to synchronize the EtherCAT master node with the TSN network and the IO nodes, i.e., the EtherCAT slave nodes, as depicted in Fig. 5c. The jitter that is caused by integrating the TSN network can be mitigated by applying a clock synchronization where the frame transmission can be scheduled and aligned by its corresponding time slot in the TSN network. It is beneficial to use the TSN clock synchronization source because all the different EtherCAT IO slave nodes are synchronized to the same time source, i.e., the TSN grandmaster clock, without introducing any cost of an additional hardware, e.g., cost of EL6688 time synchronization modules[7], apart from the network interface which was already necessary to benefit from the characteristics of TSN.

Note that the network configuration in which the TSN network resides between the master node and the slave nodes of the EtherCAT network is not the only possible architecture. In fact, our proposed solution for clock synchronization allows that the TSN network be connected at any point of the EtherCAT network. However, the configuration that is presented in this section offers many benefits. The main motivation for this configuration is that many different communication protocols might be used in industry and multiple EtherCAT master

[7]https://www.beckhoff.com/english.asp?ethercat/el6688.

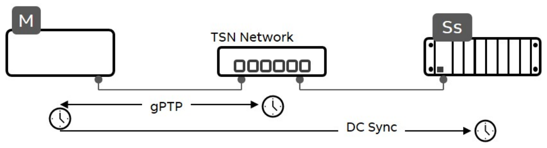


Fig. 6: Clock synchronization protocols and hierarchy in an integrated TSN-EtherCAT network.

nodes might be used. Therefore, this configuration allows us to connect the master nodes to all devices regardless of the communication protocol via a direct link to the TSN network. Thanks to this configuration we can decrease the number of connectors and cables, hence reducing complexity, weight and cost and at the same time increasing the integrability between network components.

V. Proposed Solution

The objective of this solution is to integrate the clock synchronization protocols described above with a minimum number of modifications in the devices currently used. Thanks to this integration, the EtherCAT transmissions can be synchronized with the TSN network, ensuring the correct operation of the former. This allows companies to start adopting TSN solutions while maintaining their legacy systems, and, at the same time, providing benefits of the combined network.

A. Design

The proposed solution is based on an EtherCAT master node as a reference clock of the DC synchronization from the EtherCAT point of view and the same master node as a time-aware system (grandmaster or slave time-aware system depending on the BMCA mechanism described in Section II) from the TSN point of view. This can be achieved by using an improved network interface in the master node, that can handle gPTP together with a hardware clock to minimize the jitter in the transmission. Thanks to this approach, we can synchronize the EtherCAT master node with the TSN network and the EtherCAT slave nodes with the EtherCAT master node achieving high precision between each of the elements of the combined network. Specifically, as both TSN and EtherCAT have similar precision, in the order of hundred nanoseconds. This precision will be maintained throughout the whole system. Fig. 6 shows the explained configuration. The text on the arrows indicates the clock synchronization protocol, while the arrowheads indicate who is the master and who is the slave in that synchronization protocol (the arrowhead points to the slaves from the master). The `gPTP` arrow is bidirectional because the EtherCAT master node behaves as a TSN time-aware system. The EtherCAT master node runs all mechanisms described in Section II-B, hence it can behave as a grandmaster or slave clock. On the other hand, the `DC Sync's` arrow is unidirectional as, from the point of view of the EtherCAT network, the EtherCAT master node always behaves as a reference clock.

B. Implementation

To achieve a proper implementation of the integrated solution some aspects should be taken into account. Firstly, all

EtherCAT synchronization streams should be scheduled in advance, i.e., TSN should know properties such as period, offset, and payload of the DM_T, DM_C, DM_R and SM messages, and should have a dedicated TT queue for them. This implies to configure TSN queues and the Gate Control Lists (GCLs) offline [19]. The schedule and the dedicated TT queue is a key piece to prevent the jitter of the EtherCAT synchronization messages. However, the offline scheduling of synchronization streams is not sufficient because even if the TSN network is aware of when these specific messages are going to be transmitted, if the local time at the TSN network and the EtherCAT master node is not the same then the messages can still suffer from jitter. This can cause several erroneous behaviors in the EtherCAT network, which the details will be discussed in Section VI-C.

Secondly, as anticipated above, the EtherCAT master node should be synchronized with the TSN network before using the EtherCAT synchronization mechanism to synchronize with the EtherCAT slave nodes. If this order is not respected the EtherCAT delays can be wrongly calculated. As pointed out in Section II-A, the DM mechanism relies on the symmetric propagation of messages. If the EtherCAT master node is not synchronized with the TSN network before executing the DM mechanism, DM_T can be blocked by the TSN network (as we show in Section VI). However, DM_R will not be blocked because, as the transmission through the slave nodes is deterministic, once the DM_T goes through the TSN network, when DM_R comes back the TSN network will not block the message because it will be expecting it, regardless of the potential blocking that DM_T may have suffered. If DM_T can be blocked, and DM_R cannot, then the propagation delay is not symmetric and, as pointed out, the delay is wrongly calculated. Thanks to this minor change (the improved network interface in the EtherCAT master to behave both as a TSN time-aware system and as an EtherCAT reference clock), the EtherCAT network operates as if the TSN network was not there. From the clock synchronization mechanism's point of view, all delays can be calculated correctly and the SM messages will not be blocked. Hence, the clock synchronization between the EtherCAT master node and the slave nodes will be correct. Additionally, as an SM message can be used for data transmission these messages will also be transmitted as expected. Moreover, the clock synchronization between all devices in the combined network ensures a correct transmission of other types of messages used in EtherCAT as the TSN network can be scheduled to guarantee it. However, this scheduling is beyond the scope of this paper.

VI. Evaluation

In this section, we assess the correctness of the proposed clock synchronization. As a consequence, we can ensure a proper SM data transmission resulting to a correct EtherCAT data transmission given a proper TSN scheduling.

A. Uppaal *concepts*

To formally verify the correctness of the proposed solution, and to compare the behavior of a system that combines TSN and EtherCAT with and without the solution proposed in Section V, we used the Uppaal model checker. Uppaal is a tool for modeling and verification of real-time systems.

Systems in Uppaal are modeled as networks of timed automata (finite state machines extended with a special kind of temporal variables called *clocks* that progress at the same pace) [20], extended with data types like integers, arrays, etc. The automata that conform to the system are instantiations of one or more *templates*, and those are constructed by means of *locations*, *edges*, *variables*, and *clocks*. In addition, it is possible to coordinate the operation of different automata using *channels*. Channels are special variables that can force two or more automata to take a specific edge at the same time. Uppaal provides a formal query language that can be used to specify the properties that we want to check in the model. These queries have two parts: state formula and path formula. State formulae are expressions that can be true or false depending on the state of the system, understanding as state of the system the active locations of the automata plus the value of all variables and clocks at certain moment. Uppaal does an exhaustive search of all possible states and the path formula indicates, to the query, which has to be the distribution of the states at which the state formula is true in the whole state space. For example, one path formula may require the state formula to be true for the whole state space to satisfy the query.

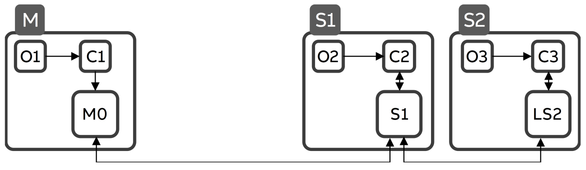
B. Uppaal *models*

We have created 3 different Uppaal models [8]. All of them modeled an EtherCAT network in which the reference clock is located in the EtherCAT master node, as described in the proposed solution in Section V. In addition, all three EtherCAT networks consist of one EtherCAT master and two slave nodes. In the following, we present a brief overview of the models and queries that are developed for evaluation purposes.

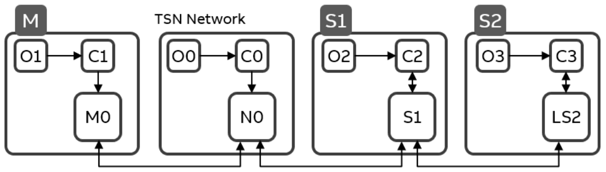
The first model M1 (Fig. 7a) consists of an EtherCAT network with one master and two slave nodes and no TSN network. All devices (master and slave nodes) include one oscillator (OX in the figure) and one local clock (CX in the figure), which operates as a simple counter only. At the beginning of the execution of the model each oscillator non-deterministically (to allow all possible combinations of choices to be checked in the state space) decides its period, which can vary between 9 and 10 time units. After that, each oscillator increases the local time of its corresponding local clock in 10 units every 9 or 10 time units, depending on the previous choice. This asymmetric evolution of local clocks emulates the drift between the local clocks. Additionally, each device has a core, which, relying on the local time provided by the local clocks, carries out the main actions. In this figure the core templates are MX, SX and LSX, which carry out the actions corresponding to the EtherCAT master, slave and last slave nodes respectively. We had to differentiate between slave

[8]The Uppaal models are publicly available at https://github.com/DanielBujosa/ETFA2020_TSN_EtherCAT_ClockSync.git.

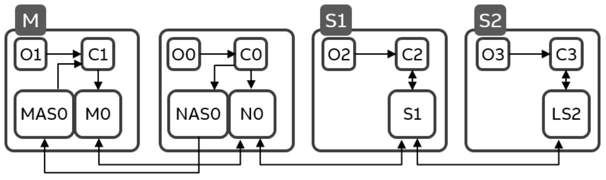
and last slave nodes because intermediate slave nodes in the network chain just need to receive and forward the messages, the last slave node is responsible of receiving the message and forwarding the response.



(a) UPPAAL M1 model.



(b) UPPAAL M2 model.



(c) UPPAAL M3 model.

Fig. 7: TSN-EtherCAT clock synchronization integration UPPAAL models.

The second model M2 (Fig. 7b) consists of the same EtherCAT network as in the M1 model but adding a TSN network between the EtherCAT master and the two slave nodes. The TSN network also presents an oscillator (O0) and a clock (C0) but its core template, represented as N0 in the figure, carries out the main actions corresponding to the TSN network. In this model, these actions consist of receiving the EtherCAT messages and forwarding them at its corresponding time slot, according to local time provided by the clock. In this case, the clock synchronization mechanism of the TSN network (gPTP) is not integrated with the clock synchronization protocol of the EtherCAT network.

The third model M3 (Fig. 7c) consists of the same EtherCAT and TSN networks as the M2 model but including the mechanism used to synchronize the EtherCAT master node with the TSN network, as explained in Section V. This mechanism is implemented in two new entities called MASX and NASX in the EtherCAT master node and the TSN network respectively. These entities provide a gPTP clock synchronization between both devices.

C. UPPAAL *queries*

In this paper we have analyzed 3 behaviors that are key to ensure good clock synchronization in a network that combines EtherCAT and TSN sub-networks.

The first behavior investigated was whether the messages used in the clock synchronization mechanisms described in Section II-A can be blocked by the TSN network. It is very important to check this behavior for two reasons. From the point of view of the delay measurement mechanism, it is important for the transmission delay to be always almost the same, both for the DM_T message and for the DM_R message. If the transmission delay changes, then the calculation of delays in EtherCAT might be incorrect. In this way, as the DM_R message cannot be blocked since the TSN network is scheduled to be in the time slot corresponding to the response once the DM_T message has been sent, as explained in Section V, if the DM_T message can be blocked, there will be a difference in the transmission delay and, therefore, the delay measurement will not be carried out correctly. On the other hand, from the point of view of the clock synchronization mechanism, if the SM message can be blocked, and also the delay is wrongly measured, the clock synchronization precision drops greatly.

The second and third behaviors we checked were the delays calculated by means of the delay measurement mechanism, described in Section II-A, and the maximum time difference between the clocks of the different devices in the UPPAAL model (CX) that conform the network, respectively. In both cases we compared the results obtained by the M2 and M3 models with the ones obtained by the M1 model. Thus, we could measure the impact of combining a TSN and EtherCAT network with and without the solution proposed in this paper.

D. *Results*

Here we present the results obtained once the above tests have been carried out.

By checking if the messages used in the clock synchronization mechanism described in Section II-A can be blocked, we determined that all synchronization messages transmitted by the master node in the M2 model may be blocked by the TSN network while in the M3 model none of the synchronization messages can be blocked. These potential blocks, as explained before, can cause problems both measuring the delays between the master node and each of the slave nodes, and may also interfere in the correct clock synchronization of them. To verify this, we obtained the delays measured in the M2 and M3 models and compared them with those obtained in the M1 model. Moreover, we did the same for the difference between the clocks, i.e., we measured the maximum difference between the different clocks of the system in the M2 and M3 models and we compared them with those obtained by the M1 model.

TABLE I: Delay measurement comparison.

Compared models	**Delay S1**	**Delay S2**
No TSN (M1) vs No Sync TSN (M2)	100%	67%
No TSN (M1) vs Sync TSN (M3)	25%	0%
Improvement M3 vs M2	x4	x$^{67}/0$

In both Table I and Table II, the first row shows the difference between values measured in the M2 model and the M1 model as a percentage, while the second row does the same with respect to the M3 model. This value is calculated by dividing the absolute value of the subtraction of the results obtained in the corresponding models by the result obtained by the M1 model. On the other hand, the third row show the improvement that the M3 model supposes with respect to the M2 model. This value is calculated by dividing the percentage obtained in the first row by the one obtained in the second row.

TABLE II: Clock difference comparison.

Compared models	**M-S1** clock diff	**M-S2** clock diff	**S1-S2** clock diff
No TSN (M1) vs No Sync TSN (M2)	200%	217%	171%
No TSN (M1) vs Sync TSN (M3)	60%	33%	29%
Improvement M3 vs M2	x3.3	x6.5	x6

As it can be seen in the third row of Table I, the M3 model, which implements the proposed solution in this paper, is at least 4 times more precise than the M2 model carrying out the delay measurement. Moreover, as it can be seen in the third row of Table II, the M3 model present a precision in the clock synchronization between EtherCAT clocks that is at least 3 times better than what is achieved by the M2 model.

In the second row of the tables we can see that there is a difference between the M1 model and the M3 model. The maximum difference shown in Table I is 25%, whereas in Table II it is 60%. However, these high values are due to the abstractions applied to the model, which are described in Section VI-B. As we had to increase the variation of the clocks to 10% and reduce the periods by several orders of magnitude, all the variations were greatly increased.

On the other hand, as an additional aspect, we tried to measure the maximum difference between the TSN grandmaster clock and the EtherCAT reference clock. However, we could not find a maximum value for the M2 model. That is an expected result because, as there is no integration between the EtherCAT and the TSN clock synchronization mechanisms, both clocks can drift indefinitely.

VII. Conclusions

TSN has shown potentials to be a promising technology for future industrial communication systems thanks to its features, such as support for mixed hard and soft real-time communications, flexibility of the traffic requirements and fault tolerance mechanisms. For this reason, industries have shown interest to adopt the TSN technology. However, they can encounter various obstacles, one of those being the legacy system support. Therefore, in this paper, we analyzed the integrability of TSN with EtherCAT, a protocol widely used in the automation domain today. We proposed a clock synchronization mechanism based on the TSN standards to achieve a high synchronization precision among EtherCAT nodes. We showed that the proposed clock synchronization mechanism is an essential component for a correct behavior of the network with respect to data transmission. We formally verified the correctness as well as the precision of the proposed mechanism based on a formal verification framework using UPPAAL tool. According to our verification, the integrated EtherCAT-TSN network with the proposed clock synchronization mechanism obtains at least 3 times higher synchronization precision among the nodes compared to not using any mechanism. The future work aims at performing an experimental implementation of the proposed solution in order to evaluate the solution's performance apart from the correct operation demonstrated in this paper.

Acknowledgments: This research has been performed with the support from the Swedish Knowledge Foundation (KKS) under the FIESTA project and the Swedish Governmental Agency for Innovation Systems (VINNOVA) under the DESTINE project.

References

[1] D. Jansen and H. Buttner, "Real-time ethernet the EtherCAT solution," *Computing Control Engineering Journal*, vol. 15, no. 1, pp. 16–21, 2004.

[2] G. Alderisi, G. Patti, and L. Lo Bello, "Introducing support for scheduled traffic over IEEE audio video bridging networks," in *Conf. Emerging Technologies Factory Automation*, 2013.

[3] K. S. Umadevi and R. K. Sridharan, "Multilevel ingress scheduling policy for time sensitive networks," in *Int. Conf. Microelectronic Devices, Circuits and Systems*, 2017.

[4] S. Kehrer, O. Kleineberg, and D. Heffernan, "A comparison of fault-tolerance concepts for IEEE 802.1 Time Sensitive Networks (TSN)," in *IEEE Emerging Technology and Factory Automation*, 2014.

[5] F. A. R. Arif and T. S. Atia, "Load balancing routing in time-sensitive networks," in *Int. Scientific-Practical Conference Problems of Infocommunications Science and Technology*, 2016.

[6] G. Cena, S. Scanzio, A. Valenzano, and C. Zunino, "Performance evaluation of the EtherCAT distributed clock algorithm," in *IEEE Int. Symposium on Industrial Electronics*, 2010, pp. 3398–3403.

[7] X. Wu and L. Xie, "End-to-end delay evaluation of industrial automation systems based on EtherCAT," in *IEEE Conf. Local Computer Networks (LCN)*, 2017, pp. 70–77.

[8] V. Q. Nguyen and J. W. Jeon, "EtherCAT network latency analysis," in *Int. Conf. Computing, Communication and Automation (ICCCA)*, 2016, pp. 432–436.

[9] H. Yi and J. Y. Choi, "Performance analysis of Linux-based EtherCAT DC synchronization," in *IEEE Int. Conf. Advanced Intelligent Mechatronics (AIM)*, 2015, pp. 549–552.

[10] G. Cena, I. C. Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "Evaluation of EtherCAT distributed clock performance," *IEEE Trans. Industrial Informatics*, vol. 8, no. 1, pp. 20–29, 2012.

[11] S. Park, H. Kim, H. Kim, C. N. Cho, and J. Choi, "Synchronization improvement of distributed clocks in EtherCAT networks," *IEEE Communications Letters*, vol. 21, no. 6, pp. 1277–1280, 2017.

[12] R. Reimann, W. Holzke, S. Menzel, and B. Orlik, "Synchronisation of a distributed measurement system," in *Int. Exhibition and Conf. for Power Electronics, Intelligent Motion, Renewable Energy and Energy Management (PCIM)*, 2019, pp. 1–8.

[13] J. Liu, L. Yang, D. Xu, and X. Wu, "A high precision clock synchronization algorithm for the EtherCAT," in *IEEE Conf. Industrial Electronics and Applications (ICIEA)*, 2017, pp. 1369–1374.

[14] B. Li, H. Lin, S. Sun, and L. Zheng, "A synchronization method for local applications of EtherCAT master-slave in Open CNC system," in *IEEE Int. Conf. Information and Automation (ICIA)*, 2018, pp. 527–533.

[15] M. Cereia, I. C. Bertolotti, and S. Scanzio, "Performance of a real-time EtherCAT master under Linux," *IEEE Trans. Industrial Informatics*, vol. 7, no. 4, pp. 679–687, 2011.

[16] R. Delgado and B. W. Choi, "On the in-controller performance of an open source EtherCAT master using open platforms," in *Int. Conf. Ubiquitous Robots and Ambient Intelligence (URAI)*, 2017, pp. 744–748.

[17] I. Song, Y. Jeon, J. Kim, S. Seo, K. Kwon, J. Chun, and J. Jeon, "Implementation and analysis of the embedded master for EtherCAT," in *ICCAS 2010*, 2010, pp. 2418–2422.

[18] D. Orfanus, R. Indergaard, G. Prytz, and T. Wien, "EtherCAT-based platform for distributed control in high-performance industrial applications," in *IEEE Conf. Emerging Technologies Factory Automation (ETFA)*, 2013, pp. 1–8.

[19] "IEEE standard for local and metropolitan area network–bridges and bridged networks," *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*, pp. 1–1993, 2018.

[20] G. Behrmann, A. David, and K. G. Larsen, *A Tutorial on* UPPAAL. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 200–236.