Mälardalen University Press Licentiate Theses
No. 338

ENHANCING TSN ADOPTION BY INDUSTRY:

TOOLS TO SUPPORT MIGRATING ETHERNET-BASED LEGACY NETWORKS INTO TSN

Daniel Bujosa Mateu

2023

School of Innovation, Design and Engineering

ISBN 978-91-7485-586-9
ISSN 1651-9256
Printed by E-Print AB, Stockholm, Sweden

“The improvement of understanding is for two ends:
first, our own increase of knowledge;
secondly, to enable us to deliver that knowledge to others."
John Locke

Acknowledgements

I am deeply grateful to my supervisors Mohammad, Julián, Alessandro, and Thomas for providing me with the opportunity to grow and develop as a researcher. Their guidance, support, and encouragement throughout the Licentiate have been invaluable, and I appreciate the effort they invested in making the journey enjoyable and productive.

I also want to express my heartfelt thanks to my family and friends, particularly my parents and brother, for their constant support and love, despite the distance between us. Their unwavering belief in me has been a driving force throughout my academic journey.

Last, but no least, I am deeply thankful to my fiancée for her invaluable support and understanding. She has been a constant source of motivation and has made this journey an unforgettable experience.

This research is supported by the Swedish Knowledge Foundation (KKS) under the FIESTA project and the Swedish Governmental Agency for Innovation Systems (VINNOVA) under the DESTINE and PROVIDENT projects. Julián Proenza was supported by Grant pid2021-124348ob-i00, funded by MCIN/AEI/ 10.13039/501100011033 / ERDF, EU.

Daniel Bujosa Mateu
April 2023
Västerås, Sweden

Abstract

New technologies present opportunities and challenges for industries. One major challenge is the ease, or even feasibility, of its adoption. The Time-Sensitive Networking (TSN) standards offer a range of features relevant to various applications and are key for the transition to Industry 4.0. These features include deterministic zero-jitter, low-latency data transmission, transmission of traffic with various levels of time-criticality on the same network, fault tolerance mechanisms, and advanced network management allowing dynamic reconfiguration.

This thesis aims to develop tools and mechanisms that enable the industry to adopt TSN easily and efficiently. Specifically, we facilitate the migration of legacy networks to TSN, enabling the preservation of most of the legacy networks and solutions while reducing costs and adoption time. Firstly, we introduce LETRA (Legacy Ethernet-based Traffic Mapping Tool), a tool for mapping Ethernet-based legacy traffic to the new TSN traffic classes. Secondly, we develop HERMES (Heuristic Multi-queue Scheduler), a heuristic Time-Triggered (TT) traffic scheduler that can meet the characteristics of legacy networks and provide quick results suitable for reconfiguration. Thirdly, we develop TALESS (TSN with Legacy End-Stations Synchronization), a mechanism to avoid adverse consequences caused by the lack of synchronization between legacy networks and TSN. Finally, we improve the Stream Reservation Protocol (SRP) to enhance Audio-Video Bridging (AVB) traffic configuration in terms of termination and consistency.

Sammanfattning

Uppfinningen av ångmaskinen i slutet av 1700-talet markerade början på en kontinuerlig och snabb process av automatisering och förbättring inom industrin, som tog ytterligare fart i och med införandet av datorstyrda maskiner och robotik i mitten av 1900-talet. Moderna fabriker och deras produkter är beroende av hundratals specialiserade processorer, inklusive sensorer, ställdon och styrenheter, som samarbetar för att utföra uppgifter och tillhandahålla tjänster. Dessa processorer är beroende av kommunikations-subsystem för att samordna och dela resurser. Dessa system, som vanligtvis kallas distribuerade system, omger oss och används av de flesta människor hela tiden. Exempel på distribuerade system finns i en mängd olika exempel, från moderna bilar till fabriker som producerar olika varor. I en bil finns det till exempel många olika sensorer och ställdon som hastighetsmätare, positionssensorer, bränsleinsprutare och tändspolar, som alla arbetar tillsammans för att se till att bilen fungerar säkert och effektivt, medan det i fabriker används robotarmar, transportband och andra anordningar för att automatisera produktionsprocessen och öka effektiviteten.

Den snabba utvecklingen av tekniken kan dock göra det svårt för företag att hålla jämna steg med de senaste verktygen och systemen, eftersom kostnaden för att införa tekniken kanske inte är kostnadseffektiv, inte bara på grund av att tekniken måste förvärvas, utan också på grund av de förändringar som införandet kräver i andra system som samarbetar. Till exempel skulle införandet av ett nytt kommunikations-subsystem kräva att alla enheter som använder det anpassas. Dessutom kräver uppgraderingen till nyare teknik ofta betydande resurser, inte bara ekonomiska utan även naturresurser. Detta kan leda till ökat avfall och ökade koldioxidutsläpp, vilket utgör en risk för miljön. Dessutom kan följderna av teknikuppgraderingar, till exempel bortskaffande av föråldrad utrustning och produktion av e-avfall, ha ytterligare miljöpåverkan.

I den här avhandlingen fokuserar vi på Time Sensitive Networking (TSN), en ny kommunikationsstandard med betydande fördelar för den framväxande tekniken. Även om TSN-tekniken ger många fördelar, bland annat högre kom-

munikationshastighet och lägre latenstider, saknar många nuvarande industrisystem mjuk- och hårdvarukraven för att stödja denna teknik. Målet med vår forskning är därför tvåfaldigt: för det första att förbättra TSN's mekanismer för att göra den mer attraktiv för industrin och för det andra att utveckla verktyg som möjliggör en sömlös migration och integration av äldre system till TSN, så att slutstationerna kan utnyttja fördelarna med TSN utan att behöva byta ut eller uppgradera större delen av systemet. Detta tillvägagångssätt sparar värdefull tid och resurser och minskar det avfall som uppstår under processen.

List of Publications

Papers included in this thesis[1]

Paper A: Daniel Bujosa, Mohammad Ashjaei, Alessandro V. Papadopoulos, Julián Proenza, Thomas Nolte. "*LETRA: Mapping Legacy Ethernet-Based Traffic into TSN Traffic Classes.*" In the 26^{th} IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2021).

Paper B: Daniel Bujosa, Mohammad Ashjaei, Alessandro V. Papadopoulos, Julián Proenza, Thomas Nolte. "*HERMES: Heuristic Multi-queue Scheduler for TSN Time-Triggered Traffic with Zero Reception Jitter Capabilities.*" In Proceedings of the 30^{th} International Conference on Real-Time Networks and Systems (RTNS 2022).

Paper C: Daniel Bujosa, Daniel Hallmans, Mohammad Ashjaei, Alessandro V. Papadopoulos, Julián Proenza, Thomas Nolte. "*Clock Synchronization in Integrated TSN-EtherCAT Networks.*" In the 25^{th} IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2020).

Paper D: Daniel Bujosa, Mohammad Ashjaei, Alessandro V. Papadopoulos, Julián Proenza, Thomas Nolte. "*Improved Clock Synchronization in TSN Networks with Legacy End-Stations.*" Technical report at Mälardalen University, Sweden, pending for submission to a journal.

Paper E: Daniel Bujosa, Inés Álvarez, Julián Proenza. "*CSRP: An Enhanced Protocol for Consistent Reservation of Resources in AVB/TSN.*" In the IEEE Transactions on Industrial Informatics 17 (TII 2020).

[1]The included papers have been reformatted to comply with the thesis layout.

Other Relevant publications[2]

Paper F: Inés Álvarez, Luis Moutinho, Paulo Pedreiras, Daniel Bujosa, Julián Proenza, Luis Almeida. "*Comparing admission control architectures for real-time Ethernet.*" In the IEEE Access, 2020, vol. 8.

Paper G: Daniel Bujosa, Mohammad Ashjaei, Alessandro V. Papadopoulos, Julián Proenza, Thomas Nolte. "*Work-in-Progress: The Effects of Clock Synchronization in TSN Networks with Legacy End-Stations.*" In the 27^{th} IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2022).

[2]Not included in this thesis.

Contents

I

Thesis

Chapter 1

Introduction

The emergence of novel technologies presents potential solutions and enhancements to industries that can confer a competitive edge by minimizing costs, improving products, or promoting environmental sustainability. By leveraging such technologies, businesses can increase performance, optimize resource utilization, and reduce harmful emissions. Nevertheless, embracing these opportunities is not without obstacles. The challenges often stem from the practicality and feasibility of integrating the new technologies into the industry.

Time-Sensitive Networking (TSN) is a new technology that has the potential to reshape industrial communications and facilitate the transition to Industry 4.0. The development of TSN can be traced back to the creation of the IEEE Audio-Video Bridging (AVB) Task Group (TG) in 2005. This group focused on adding real-time capabilities to Ethernet for audio and video streaming. The AVB TG developed three projects: IEEE Std 802.1AS [5] for clock synchronization, IEEE Std 802.1Qav [1] for Credit-Based Shaping (CBS), and IEEE Std 802.1Qat [2] for the Stream Reservation Protocol (SRP). To ensure minimum Quality of Service (QoS) when using the aforementioned standards, the AVB TG also created a set of rules called IEEE Std 802.1BA-2011: Audio Video Bridging Systems [3]. Together, these standards are known as the AVB standards.

As interest in AVB technology grew beyond audio and video streaming, the AVB TG was renamed to TSN TG in 2012[1]. The TSN standards are an expansion of the AVB standards to meet the needs of additional applications, such as automotive[49], automation[59], and energy distribution[48]. TSN offers a variety of compelling features, including support for mixed hard and soft real-time communications, flexibility of traffic requirements, and fault tolerance mechanisms. These features enable TSN to provide new solutions within

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

modern industrial systems, such as increased bandwidth, improved real-time behavior, improved fault tolerance, and even the integration of multiple legacy networks into a single TSN network.

However, most Ethernet-based communication networks nowadays utilize a variety of devices and protocols that cannot be seamlessly integrated with TSN. Additionally, due to hardware limitations, several of these networks would not be able to support TSN integration. Therefore, the adoption of TSN by the industry requires significant investments in both software and hardware upgrades. Such modifications often result in time and resource costs that may not be cost-effective. Furthermore, the upgrade process leads to the generation of a substantial amount of technology waste, consisting of relatively good-condition equipment that could have been reused.

In this work, our objective is to develop tools and mechanisms to facilitate the migration and integration of legacy networks with TSN. This integration can be done in different ways, such as through the use of gateways. However, this would not allow the legacy networks to take advantage of other TSN features such as higher bandwidth or low jitter. Thus, we propose to directly replace the communication subsystem of the legacy network, i.e., all devices exclusively responsible for communication (excluding the end-stations), with TSN. This integration technique ensures that legacy end-stations can keep their communication behavior and protocols, agnostic of the change, leading to improved integration of various legacy networks and enabling them to leverage the advantages of TSN potential features.

In order to achieve the desired migration and integration of legacy networks with TSN, several challenges should be tackled. Firstly, legacy traffic needs to be identified and defined accurately. Secondly, the legacy traffic should be mapped into the different TSN traffic types, based on its specific requirements. Finally, the traffic mapped as TSN TT traffic should be scheduled accordingly. Furthermore, if scheduling is required, it is necessary to have a global view of time, and thus proper synchronization between the legacy network and TSN must be ensured.

In this thesis we address the above challenges by proposing novel and efficient solutions. Firstly, we identify Ethernet-based trafic parameters relevant for mapping legacy traffic into TSN. Secondly, we propose a mapping algorithm that can efficiently map Ethernet-based traffic into TSN trafic classes by taking into account their timing requirements. Finally, we develop a TT traffic scheduling algorithm that conforms to the requirements of legacy networks and a synchronization mechanism that enables the communication of legacy scheduled traffic even when the legacy end-stations and TSN are not synchronized.

Outline. The Licentiate thesis is organized as follows. Chapter 2 provides the necessary background for the understanding of this work. Chapter 3 presents the problems identified and addressed in the thesis. Chapter 4 presents the related work. Chapter 5 describes the research method used in the work. Chapter 6 lists the challenges and contributions covered in this thesis and, finally, Chapter 7 concludes the paper and presents future directions.

Chapter 2

Background

This chapter summarizes the main mechanisms involved in the work and the background necessary to facilitate its understanding.

2.1 TSN traffic classes

TSN end-stations communicate by transmitting Ethernet frames through routes consisting of links and TSN switches. TSN end-stations and switches are nodes that support clock synchronization and traffic shaping. TSN devices feature output ports that support up to eight First-In-First-Out (FIFO) queues, each associated with a specific priority level. TSN supports three traffic classes: Time-Triggered (TT) traffic, Audio-Video Bridging (AVB) traffic, and Best-Effort (BE) traffic. Each priority level is assigned to one of these traffic classes depending on the shaping mechanisms it applies. However, it is common that TT traffic has the highest priority, while BE traffic has the lowest priority. It is important to note that multiple queues can cover the same traffic class. For example, AVB traffic can consist of classes A, B, and C, each associated with a distinct priority level.

Figure 2.1 shows an example of a TSN device output port with four queues configured as TT traffic with the highest priority, AVB classes A and B traffic with the second and third highest priority, and a BE traffic class as the lowest priority. This is determined by the mechanisms applied to each queue, including Time-Aware Shaper (TAS) and CBS, which will be described in depth below.

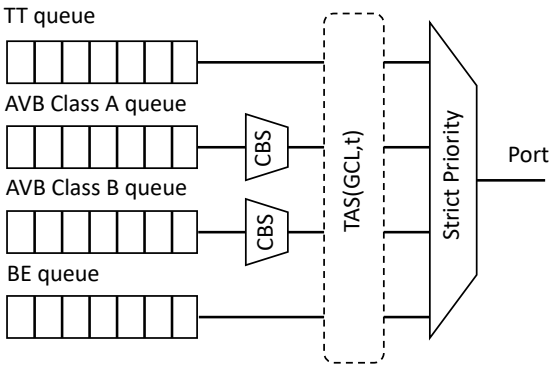


Figure 2.1: A TSN output port with four FIFO queues: one TT queue, two AVB queues, and one BE queue.

2.1.1 Time-Triggered Traffic

TT traffic is transmitted in accordance with a fixed offline schedule that specifies the exact time slot each TT frame is transmitted. To prevent interference between frames, TT traffic uses the TAS mechanism shown in Figure 2.1 as defined in IEEE 802.1Qbv [30]. The TAS mechanism associates each queue with a gate that can be opened or closed. Frames in a queue can be transmitted only when the gate is open; otherwise, they are blocked. The Gate Control List (GCL) is responsible for managing the gates, specifying precisely when they should open and close. The GCL is a cyclic list that repeats the schedule, with each entry in the list specifying the precise time at the nanosecond level when the gate should be open or closed. The term *transmission window* refers to the time interval during which a gate is open.

Figure 2.2 depicts an example of how the TAS operates for two TT queues with different priorities (6 and 7). The example assumes the transmission of three TT frames with a period of 4 time units through a switch port. One of the frames is set to the highest priority 7 (blue), while the other two frames are set to priority 6 (red and green). The figure shows the gates for the two queues and their states at different time slots based on the GCL, which specifies when the gates should be open or closed. During the first time slot (T0 to T1), the gate for the priority 6 queue is open (1 in GCL), while the gate for priority 7 is closed (0 in GCL), allowing the transmission of the red frame. In the second time slot (T1 to T2), the gate for priority 7 is open, allowing the transmission of the blue frame. Both gates are closed in the third time slot (T2 to T3), preventing any transmission. Finally, in the last time slot, the

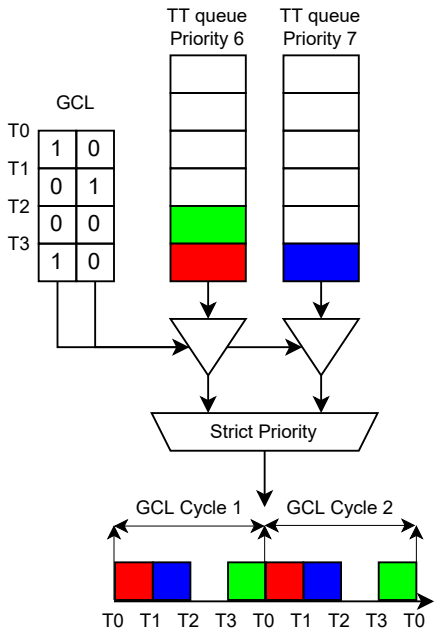


Figure 2.2: TSN TAS gate mechanism.

gate for the priority 6 queue is open, allowing the transmission of the green frame. The figure also shows two cycles of frame transmission at the bottom, demonstrating how the GCL repeats the schedule.

2.1.2 Audio-Video Bridging Traffic

The AVB TG [1] presented the CBS that applies credits to AVB queues. Credits are consumed when a frame in that queue is transmitted, and are replenished when there is a pending frame in the queue or if the credit is negative, even if no frame is waiting in the queue. These consumption and replenishment ratios, which are configured in the CBS, are constant and determine the bandwidth reserved for each AVB queue. To transmit data, AVB queues need to have a positive or zero credit, and their gate must be open based on the TAS and GCL. CBS usually separates classes into A and B, and allows lower-priority traffic transmission even if higher-priority traffic is waiting, based on the credits. This leads to improved QoS for lower-priority traffic and reduced buffering. Despite the unknown activation time of AVB traffic due to potential blocking from other AVB classes or ST queues, there are techniques to estimate its worst-case response time [11].

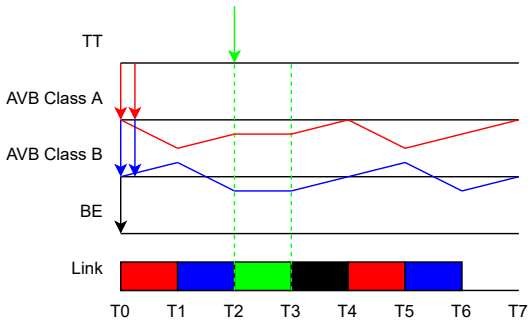


Figure 2.3: TSN CBS mechanism.

The operations of CBS for two AVB queues (Classes A and B) interacting with one higher priority TT queue and one lower priority BE queue are illustrated in Figure 2.3. The downward arrows in the figure indicate when a frame arrives at its respective queue of an output port, the vertical dashed lines indicate the TT traffic window, and the slopes indicate the credit evolution. The figure demonstrates that at T0, two frames of each AVB class and one BE frame are ready to be transmitted. As the credit of both AVB queues is zero, class A begins transmitting the first frame because it has higher priority. This consumes its credit while increasing the credit of AVB class B. By T1, the credit of AVB queue class A becomes negative, causing class B to begin transmitting, even though a higher priority frame is waiting. At T2 a frame arrives at the TT queue as scheduled, hence GCL closes all gates but the corresponding to the TT queue to ensure its transmission without interruptions. Note that, during the transmission of TT traffic, the credits are frozen, i.e. not changing. After transmitting the TT frame at T3, the AVB traffic credit remains negative, allowing the BE frame to be transmitted even though two higher priority frames are waiting. Finally, the credits are replenished during the transmission of the BE frame. Therefore, at T4 and T5, AVB frames of classes A and B are transmitted as in T0 and T1.

2.1.3 Best-Effort traffic

The lowest priority traffic type, BE, does not offer any real-time guarantees. A queue that carries BE traffic is not shaped by CBS and can only transmit frames if its gate is open, and all other AVB queues have negative credit or if there is no AVB traffic available for transmission. This behavior is illustrated in Figure 2.3 at time T3 to T4 where the credits for both classes A and B are negative.

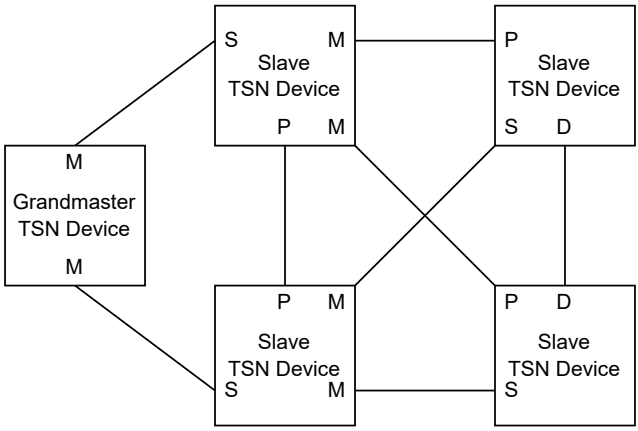


Figure 2.4: Example of TSN time-synchronization spanning tree.

2.2 TSN Clock Synchronization

The IEEE 802.1AS standard [5] describes the gPTP mechanism that provides TSN clock synchronization, which is composed of three key parts: the Best Master Clock Algorithm (BMCA), the Propagation Delay Measurement (PDM) mechanism, and the Transport of Time-synchronization Information (TTI). The BMCA determines the grandmaster clock, which acts as the reference clock in the TSN network, and establishes the hierarchy between TSN devices (TSN end-stations and switches). After establishing the hierarchy, the PDM mechanism is used to measure the propagation delay between TSN devices. Finally, the TTI mechanism is used to disseminate the grandmaster time to synchronize the other TSN devices. The following subsections provide a detailed description of each of the three mechanisms.

2.2.1 BMCA

The BMCA algorithm constructs a spanning tree for time synchronization, with the grandmaster TSN device as the root. An example of this, is illustrated in Figure 2.4. In this tree, each TSN device can act as either a grandmaster or a slave, and each port can be categorized as a Master port (M), Slave port (S), Passive port (P), or Disabled port (D). To determine these behaviors, each system periodically broadcasts a special message called `announce message`. This message contains various parameters, but we focus on two parameters, the `systemIdentity` and the `stepsRemoved`. The `systemIdentity` parameter indicates the accuracy of the sender's

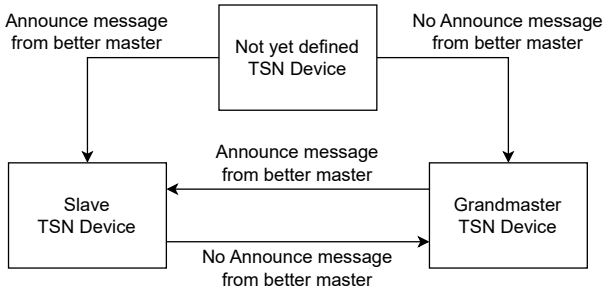


Figure 2.5: Time-aware system BMCA evolution.

clock, while `stepsRemoved` parameter denotes the distance between the transmitter and the receiver. Specifically, the `stepsRemoved` value is incremented each time the `announce message` is forwarded. For instance, consider a line topology consisting of three TSN devices, where the first device sends its `announce message`. The second device receives this message with the `stepsRemoved` parameter value of 0, but before forwarding it to the next device, it increments the `stepsRemoved` value. Thus, the last TSN device receives the `announce message` sent by the first device with a `stepsRemoved` parameter value of 1.

The diagram in Figure 2.5 illustrates how a TSN device can either act as a slave or grandmaster clock if it does not have an assigned role. A TSN device becomes a slave if it receives an `announce message` from a better clock, i.e., a message with a greater `systemIdentity` parameter. Conversely, if the TSN device does not receive any `announce message` from a better clock within a defined period (defined by the periodicity of the `announce message` transmission), it becomes the grandmaster clock. Similarly, a grandmaster or a slave TSN device can switch roles based on the reception of an `announce message` from a better clock or lack of it. On the other hand, the proximity to the grandmaster determines the roles of ports in a TSN device, which can be determined using the `stepsRemoved` parameter. The port closest to the grandmaster becomes the slave port, and only one port in the TSN device can have this role. The port closest to the grandmaster clock in a link becomes the master port. Disabled ports are those that are explicitly disabled, while ports that are neither master, slave, nor disabled are passive ports.

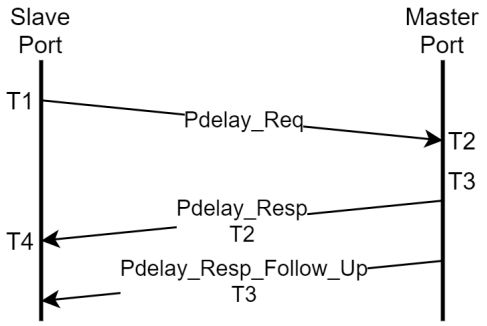


Figure 2.6: PDM diagram.

2.2.2 PDM

Once the spanning tree has been created with the grandmaster as the root, PDM is used by the slaves to calculate the propagation delays between its slave port and the master port of the TSN device connected to it. The process is illustrated in Figure 2.6. The PDM process begins with one slave transmitting a delay request message `Pdelay_Req` via its slave port to another TSN device, which may be the grandmaster or another slave. The initiating slave records the time at which the message is sent ($T1$). The receiving TSN device receives the message through its master port, records the time at which the message is received ($T2$), and transmits $T2$ back to the initiating slave while recording the transmission time ($T3$). The initiating slave receives $T2$ and records the time at which it is received ($T4$). Finally, the receiving TSN device transmits $T3$ to the initiating slave, allowing the latter to calculate the delay as $Delay = \frac{(T4-T1)-(T3-T2)}{2}$.

2.2.3 TTI

After the creation of the spanning tree with the grandmaster as the root and the measurement of the delays by the slave TSN devices, TTI is initiated. TTI involves the TSN devices transmitting their local time via their respective master ports to the slave TSN devices connected to them. The TSN devices, which receive the message via their slave ports, add the previously measured delay and update their local time accordingly.

2.3 Stream Reservation Protocol

The TSN TG relies on SRP in many of its projects, as it plays a crucial role in verifying resource availability within the network and reserving these resources, which enables a bounded end-to-end delay and prevents packet loss caused by the buffer overflow. Furthermore, the flexibility of SRP allows for dynamic traffic requirement modifications during run-time. While there exist three different SRP architectures, this work will focus solely on the distributed version. For additional information regarding the other architectures, please refer to [4].

SRP follows the publisher-subscriber paradigm to enable real-time data communications through streams. In this paradigm, the publisher, known as a *talker*, transmits data to subscribers, known as *listeners*. Each stream in SRP is a logical communication channel that carries traffic defined by a set of parameters such as the frame size and period. For example, if a temperature sensor acts as a talker and wishes to transmit its measurements with a 10 ms period and a 1-byte payload to other end-station (the listeners), the network must first verify that sufficient resources are available. If that is the case, the network will then create a stream with the specified period and payload to transmit the sensor data.

It should be emphasized that decisions regarding the reservation of resources are solely made based on local information. However, there is crucial information related to the reservations that need to be distributed throughout the network. For instance, this information may include the amount of resources required for a stream or whether a particular switch has sufficient resources available. Such information is transmitted via specialized messages referred to as talker and listener attributes.

An example of the SRP mechanism in a linear network topology is described in Figure 2.7. The network comprises a talker (T), a listener (L), and two switches (S1 and S2). Before transmitting frames, the talker must create a stream by broadcasting a Talker Advertise (TA) message. This message includes stream identification and resource requirements which are used in the Admission Control (AC) by the rest of the devices of the network to check whether there are enough resources for the stream to be created. Switches receiving the message check output ports for resource availability. If an output port has insufficient resources, it sends a Talker Failed (TF) message with the reason for failure. If the output port has enough resources, it forwards the TA message to the next device.

End-stations that are not interested in the stream do not take any further action on receiving the TA or TF messages. However, if a end-station wants

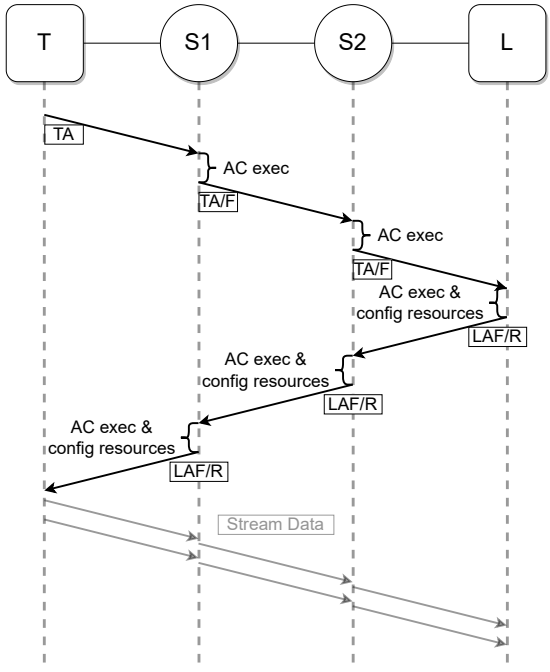


Figure 2.7: Time diagram of the resource reservation mechanism in a network with a linear topology.

to become a listener, it checks its resources and sends a Listener Ready (LR) message to the switch if it has enough resources. If it does not have enough resources, it sends a Listener Asking Failed (LAF) message. The switch receiving the LR or LAF messages checks its resources again and forwards a combined response to the talker: (i) if all ports receive LR messages, the switch transmits an LR message to the talker; (ii) if all ports receive LAF messages, it transmits another LAF message to the talker; and (iii) if there is a mix of LR and LAF messages, it transmits a Listener Ready Failed (LRF) message to the talker.

Finally, the talker waits for an LR or LRF message before starting data transmission. Additionally, it can delete the stream at any time using the unadvertise stream mechanism, which is broadcast to all devices.

Chapter 3

Problem Formulation

In this work, we aim to facilitate the adoption of TSN by the industry via developing tools and mechanisms to enable the migration and integration of legacy networks with TSN. While there are several approaches to adopt TSN, including complete replacement of the existing network with TSN software and hardware or using TSN as a backbone while connecting all legacy networks through gateways, these options have limitations. The former is a resource-intensive process, whereas the latter does not allow the legacy networks to benefit from TSN features. Therefore, we propose replacing the communication subsystems of legacy networks, i.e., the set of devices exclusively responsible for communication excluding the end-stations, with a single TSN network, while ensuring that the legacy end-stations, applications, communication protocols, and traffic with different timing requirements continue to operate as effectively as before, if not better. This approach allows the legacy end-stations to take advantage of the benefits offered by TSN, such as high bandwidth, support for mixed hard and soft real-time communications, flexibility of traffic requirements, and fault tolerance mechanisms. Our proposed method enables the migration and integration of legacy networks with TSN, thereby facilitating a smooth transition to this technology.

To enable the industry to adopt TSN solutions, and achieve the desired integration, a proper migration and integration methodology of the legacy traffic to TSN must be designed. This poses several challenges, which are addressed in this work. Regarding the traffic migration, the legacy traffic must be classified into the three previously mentioned TSN traffic classes. Such mapping must meet the timing requirements of the legacy traffic. On the other hand, regarding the legacy networks integration, the traffic classified as TT due to its high time requirements must be scheduled, considering its legacy characteristics. In addition, traffic scheduling must be scalable to allow for the migration

of large networks and must be fast in case new or legacy reconfiguration mechanisms need to be supported. Finally, the TSN schedule must be aligned with the transmission of the frames in the legacy network even when synchronization between the legacy end-stations and the TSN network is not possible.

Finally, a previous work [15] showed that SRP lacks termination and consistency, which can result in the inefficient utilization of bandwidth. This can limit the network's ability to integrate with and support legacy networks, thereby hindering its overall scalability. We will next take a closer look at each of the problems to be solved.

3.1 Legacy Ethernet-based Traffic Mapping

The mapping consists of clustering the legacy traffic based on its characteristics into the three types of TSN traffic, including TT, AVB, and BE traffic. In this thesis, we will focus on Ethernet-based legacy traffic. One of the main features of TSN, and the reason behind the growing interest in adopting it by the industry, is its traffic flexibility, i.e., its ability to combine several types of traffic on the same network. Thanks to this feature, TSN seems to be key to advancing the industry to the incipient Industry 4.0 paradigm but also makes it possible for TSN to integrate different legacy networks in the same TSN network. However, each legacy network has particular characteristics that are not directly linked to the different types of TSN traffic; hence a proper mapping methodology is key to the proper migration and integration of legacy networks into TSN. To achieve this, we must identify the characteristics of the legacy traffic that are relevant to defining its behavior in the TSN network and, based on these characteristics, split the traffic among the different types of TSN traffic classes.

3.2 TT Scheduling

As mentioned before, in TSN, a GCL is defined for each queue in each TSN device output port, in such a way that the GCL identifies the moments in which the gate of each queue will be open. The scheduling of TT traffic, and its synthesis in GCLs, is known to be an NP-complete problem [44]. Several solutions are proposed in the literature to schedule TT traffic in TSN networks that are mainly based on Integer Linear Programming (ILP) and Constrained Programming (CP) [10]. These solutions are known to have high time complexity, i.e., they require a long time to schedule large networks, thus they are not generally scalable. In addition, these solutions are not suitable for networks that

require dynamic reconfigurations as the new configuration should be created relatively fast. A few heuristic schedulers are also proposed, e.g., [40], whose performance is not properly compared with the ILP and CP solutions. In this work, we seek to develop a heuristic scheduler capable of synthesizing the GCLs of the legacy traffic mapped as TT traffic with acceptable performance and low scheduling times enabling the migration of large legacy networks.

3.3 Legacy Networks Synchronization

TSN TT traffic requires the network to be fully synchronized. Otherwise, different devices could exhibit clock drift, which would cause the transmission and reception of frames to not properly match the TSN schedule. For example, Figure 3.1 shows the effects of transmitting TT traffic between two legacy end-stations that are synchronized via a legacy clock synchronization, through a TSN network that is not synchronized with them. As we can see, the sender transmits frames faster than the TSN network forwards them in its transmission windows. This causes frames to arrive at the receiver increasingly later than their legacy scheduled time (dashed envelopes). Furthermore, since the transmission of frames by the TSN network to the listener is slower than the transmission by the talker to the TSN network, the frames stack up in the buffers. However, the buffers are not infinite, hence frames that arrive once the buffer is full are discarded. Similar phenomenon occurs when the TSN clock is faster than the legacy network one. A more detailed description of the causes and consequences of the effects of the lack of synchronization between legacy end-stations and TSN can be found in paper [16].

Despite the importance of synchronization, in many cases, legacy networks may not be able to implement TSN's synchronization protocols. Therefore, it is necessary to analyze the effects of the lack of synchronization between legacy networks and TSN in heterogeneous networks, i.e. networks combining TSN and legacy end-stations, and to develop mechanisms to avoid such adverse effects.

3.4 Distributed SRP

As mentioned before, paper [15] demonstrated the lack of termination and consistency of the distributed SRP at both the application and infrastructure levels. On the one hand, termination issues are mainly due to the fact that SRP listeners do not inform the bridges nor the talkers when they are not interested in binding to a stream. On the other hand, consistency issues are mainly due to

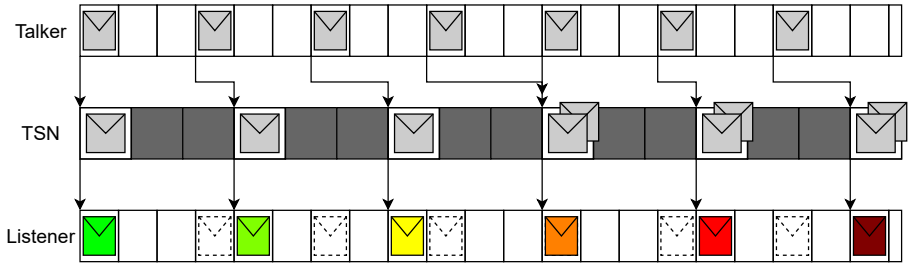


Figure 3.1: Example of positive drift synchronization issue.

the fact that information related to the reservations is propagated in a single direction. That is, the talker attribute transmitted by a talker is forwarded always towards the listeners; while, when listeners and switches reply to a stream declaration, the information is only forwarded towards the talker. Thus, not all the devices involved in the reservation of a stream receive the same information.

This does not directly affect the migration and integration of legacy traffic, but may lead to a waste of bandwidth and resources that limit both the addition of new traffic and the adoption of legacy traffic. In this regard, we need to improve the distributed SRP in order to provide network devices with a consistent view of the reservation of resources so that they can make rather complex decisions within a bounded time avoiding waste of resources..

Chapter 4

State of the Art Review

The TSN TG's work since 2012 has been highly relevant, resulting in the community dedicating a significant amount of research to the study, application, and improvement of TSN. For instance, the work in [6] studied the effects of the TAS, the work in [31] analyzed the fault tolerance issues, while the work in [7] proposed time redundancy to tolerate temporary faults. In addition, the work in [58] studied the scheduling policies, the works [61] and [35] analyzed schedulability of traffic with different TSN features and the load balancing was studied in [8]. Moreover, the work in [10] provided an up-to-date comprehensive survey of the TSN-related research.

There are also a few works on reconfiguration and integration in dynamic system networks. For example, on the one hand, work in [47] analyzes the ability of SDN to accelerate the Time-to-Integrate process in evolving topologies from a synchronization point of view. On the other hand, the works in [27] and [28] aim to provide auto-configuration to TSN by introducing a Configuration Agent into the network, an entity that continuously monitors the network for changes and automatically updates the configuration to adapt to those changes while maintaining the desired quality of service.

Other works on integrating legacy networks into TSN networks are the works in [38] and [51]. The former integrated a few of the TSN standards into Sercos III, which is a closed system that allows standard Ethernet devices to be plugged in, to improve its performance; while the later proposed an integration methodology of wireless TSN (802.11). However, as [18] shows in their experimental setup, there are still many challenges to enable a proper integration between TSN and Non-TSN devices. Therefore, in this work we focus on analyzing the works that address the problems mentioned above.

4.1 Mapping

Regarding traffic mapping, a meta-heuristic method is proposed in [23] that maps mixed-criticality applications into the TSN traffic classes. Although the aim is similar to ours, the proposed method does not cover all cases that are studied and exist in industrial applications. The method, thus, becomes suitable for cases where only very few mixed-criticality levels are assumed in the network with no extensive timing information. Other papers such as [21] only map a specific type of traffic while papers such as [19] analyze the characteristics of different types of TSN traffic and provide guidance on how to perform the mapping. However, to the best of our knowledge, there are no automated tools for mapping legacy traffic based on its characteristics into TSN.

4.2 Scheduling

Within the context of TT traffic scheduling in TSN networks, the works in [50] and [39] present a joint routing and scheduling algorithm formalized as an ILP and as a meta-heuristic scheduling approach based on a Genetic Algorithm (GA) approach, respectively. The work in [20] presents an SMT-based scheduler capable of scheduling networks with several TT queues. The work in [24] proposes a GCL synthesis approach based on Greedy Randomized Adaptive Search Procedure (GRASP) meta-heuristic [45], which takes AVB traffic into consideration, whereas the work in [25] proposes a joint routing and scheduling approach for TT and AVB traffic by means of an integrated heuristic and meta-heuristic strategy. In the latter work, the K-Shortest Path (KSP) method [60] is utilized for routing, and GRASP is used to schedule both TT and AVB at the same time. Moreover, the work in [12] synthesizes a network topology that supports seamless redundant transmission for TT traffic by proposing a greedy heuristic algorithm for joint topology, routing, and scheduling synthesis. Finally, paper [56] provides a comprehensive survey of scheduling techniques and algorithms used in TSN to support time critical applications.

The above-mentioned solutions are mostly based on ILP or constraint programming, while some of them exploit the use of meta-heuristics, e.g., GA. However, these solutions normally are highly time-complex, which makes them not scalable. Few works target heuristic solutions with lower time complexity. For instance, the work in [40] proposes a heuristic routing and scheduling algorithm called Heuristic List Scheduler (HLS) that is limited to a single TT queue, while the work in [57] compares 4 heuristic algorithms combining routing and scheduling (Modified Most Loaded Heuristic (MML), Bottleneck

Heuristic (BN), Coefficient of Variation Heuristic (CV) [9][33], and Modified Dot Product Heuristic (MDP) [41]), all with scheduling times greater than 100 ms and unable to handle multiple queues.

Despite the large amount of work done on scheduling in TSN, most works are not a priori compatible with the specific characteristics of the legacy traffic, such as offsets, drifts, or specific reception jitters. Moreover, solutions are normally time-complex or very limited in terms of schedulability.

4.3 Synchronization

One of the key features of TSN is clock synchronization. Most of the works in the literature are focused on integrating TSN with wireless or 5G networks. For example, in work [29] authors implement a low-overhead beacon-based time synchronization mechanism to provide highly accurate synchronization to the wireless networks, thus they can be used in the context of high determinism TSN networks. Moreover, works [13] and [46] extend IEEE 802.1AS and IEEE 802.11 respectively with the intention of integrating TSN with wireless networks while [52] discusses the integration challenges of Wired TSN and Wireless Local Area Network (WLAN) technologies and proposes a Hybrid TSN device architecture. On the other hand, [26] introduces the integration of TSN time synchronization (IEEE 802.1AS) conform with 5G while [17] proposes a cross domain clock synchronization method based on data packet relay to solve the end-to-end cross domain clock synchronization problems caused by the different 5G-TSN integrated network clock domains. Finally, other works such as [54], [53] and [55] has evaluated the performance of 5G-TSN networks.

According to our state of the art review, and to the best of our knowledge, there is no work addressing the challenges of synchronizing legacy devices onto a TSN network.

4.4 AVB SRP

There are many works related to the study of AVB's efficiency. For example, in [43] the authors apply network calculus to evaluate the real-time performance of Ethernet AVB in automotive networks and in [37] the authors provide insights into the performance of AVB and TSN in automotive Ethernet networks, concluding that both protocols can provide reliable and deterministic communication but their performance depends on network configuration and traffic characteristics. Moreover, paper [36] proposes an extension to the

IEEE 802.1 AVB protocol to allow for the coexistence of synchronous and asynchronous traffic and paper [34] addresses the challenges of designing an IP/Ethernet-based in-car network for real-time applications, suggesting solutions such as network segmentation and quality of service mechanisms for which AVB may be relevant. On the other hand, in the work presented in [42] the authors detect a drawback in the resource reservation de-registration specification, which leads to the waste of the network resources, and proposed some solutions. Moreover, some works present solutions to provide fault tolerance against permanent faults using SRP [32].

Nevertheless, in a previous work [15] we detected the lack of termination and consistency properties in the SRP and how this could cause bandwidth losses or loss of new requests due to overbuffering. Although we proposed some solutions, these were not implemented or evaluated in any way before.

Chapter 5

Research Methodology

The objective of this thesis is the development of methodologies capable of overcoming the challenges presented in this work and their corresponding implementation in tools and mechanisms, either independent or integrated, that facilitate the adoption of TSN by the industry. For this purpose, we followed the hypothetico-deductive [22] research method. Figure 5.1 shows the process of the research.

- **Start**: This project aims to facilitate the industry to adopt TSN both by providing tools and mechanisms for the migration and integration of legacy networks with TSN as well as by eliminating defects in existing mechanisms that could limit such migration and integration.
- **Literature Review**: Once the objective is defined, we perform an extensive state-of-the-art review on Ethernet traffic mapping, TT traffic scheduling and synchronization, and AVB traffic configuration. This review includes papers and standards with the intention of understanding the technology in depth and looking for possible points of conflict for the integration of legacy networks, as well as deficiencies in the existing protocols and mechanisms.
- **Identification of Limitations**: With the knowledge gained through the literature review, we identify the limitations of TSN in terms of migration and integration of legacy networks as well as the performance of its mechanisms. In cases where the review is not sufficient to fully identify the limitations, we raise hypotheses that we verify by running experiments and/or simulations with models.

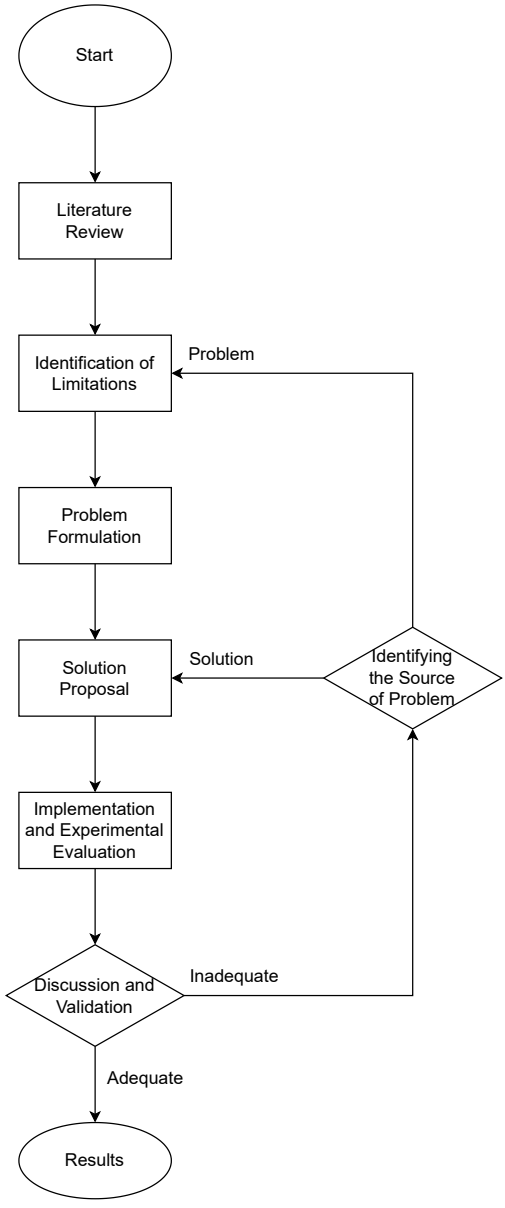


Figure 5.1: Research Methodology

- **Problem Formulation**: In this stage, we identify the problems to be solved and the objectives to address them based on the results obtained during the literature review and the identification of limitations.

- **Solution Proposal**: Once the problems to be solved have been identified and defined, novel solutions which address such problems are discussed and proposed.

- **Implementation and Validation**: The solutions from the previous stage are implemented as as prototypes, tools or models and are evaluated. The effectiveness of the solution is inspected by a feedback loop. If the evaluation output is judged inadequate, we trace back to identify the source of the problem. If the inadequate output is due to the problem formulation, the process is returned and performed from the beginning. Otherwise, the proposed solution is reassessed by refining the current solution or defining a new one. Finally, if the results of the evaluation are deemed adequate, i.e. the limitation identified through the literature review or models and/or experiments in the "identification of limitations" phase has been successfully resolved with a sufficient degree of certainty, the process ends with the publication of the proposed solution. Regarding validation, we conduct experiments for the evaluation of the tools and implemented solutions and formal verification for the validation of the models.

Chapter 6

Thesis Goals and Contributions

The overall goal of this thesis is to enable and facilitate TSN adoption by the industry. To this end, it is essential to develop tools and mechanisms that enable the migration and integration of legacy networks to TSN; otherwise, it would be necessary to replace almost the entire network, which would involve high costs and time.

6.1 Research Gaps and Industry Needs

There has been a growing interest in TSN by the industry in the last few years. This is due to its characteristics which include high bandwidth combined with real-time capabilities, traffic flexibility, and fault tolerance mechanisms, to name a few. These features are key for the integration and interoperability of different levels of operation at the industrial level which seems key for the evolution to an Industry 4.0 paradigm as well as for new products and increasingly large and complex solutions. However, implementing TSN in factories that are already established and in operation or in products may not be cost-effective as it would require changing all their existing devices, networks, and solutions. Therefore, in this work we analyze the limitations of TSN to manage the migration and integration of legacy networks and propose solutions. Looking at the current state of the art we have identified a set of key research gaps given the industry needs towards achieving the overall goal. The limitations identified through the literature review and the experiments and models during the "identification of limitations" phase were synthesized into the following research gaps:

- **RGap**$_1$: In TSN the traffic is divided into 3 types of traffic (TT, AVB, and BE) which in turn can have different priority levels (up to a maximum of 8 priority levels in total). Each of the traffic types has unique characteristics that do not need to have a direct correspondence with the legacy traffic to be migrated or integrated into the new TSN network. Inadequate traffic mapping in the new network may result in not meeting the requirements of the legacy networks.

- **RGap**$_2$: The scheduling of TT traffic, and its synthesis in GCLs, is known to be an NP-complete problem [44]. This leads to either unscalable or low-performance schedulers that, in most cases, do not take into account the specific requirements of legacy traffic.

- **RGap**$_3$: The lack of synchronization is also a problem since drift between clocks causes variations between reserved and required bandwidth. This is especially problematic in the case of TT traffic where such drift can lead to loss of frames or additional delays of up to an entire period.

- **RGap**$_4$: Finally, during the literature review and the search for limitations, we detected termination and consistency issues in the distributed version of the SRP which may lead to a waste of resources that can limit the migration and integration of legacy traffic.

6.2 Research Goals

Given the set of research gaps, we have identified 4 specific research goals of this thesis, i.e., the research goals required to solve the aforementioned problems are:

- **RG**$_1$: Develop a mapping methodology to categorize Ethernet-based traffic into the three types of TSN traffic, including TT, AVB, and BE traffic, with the goal of maximizing its schedulability.

- **RG**$_2$: Develop a scalable heuristic TSN TT traffic scheduler that can handle large volumes of traffic within a reasonable time while maintaining high schedulability using multiple TT queues.

- **RG**$_3$: Develop a synchronization mechanism that eliminates the drift between the TSN network and the legacy network schedule caused by the lack of synchronization. The mechanism should adapt the non-TT transmission windows to adjust transmission and reception ratios, thus

preventing the adverse effects caused by the drift and ensuring seamless integration in terms of synchronization.

- **RG**$_4$: Improve the distributed SRP configuration protocol to provide it with termination and consistency to ensure proper configuration of the TSN AVB traffic avoiding waste of resources.

6.3 Research Contributions

The research contributions in the thesis to address the research goals are as follows:

- **RC**$_1$: We develop a Legacy Ethernet-based Traffic model that can characterize traffic of any Ethernet-based communication protocol. The traffic model will help us to defines the behavior of the legacy traffic by means of a set of parameters that will later be used to map it. This model will partially address RG$_1$.

- **RC**$_2$: We develop a mapping methodology that can map the legacy Ethernet-based frames characterized by the model proposed in RC$_1$ into different TSN traffic classes. We implement the mapping method as a tool, named Legacy Ethernet-based Traffic Mapping Tool or LETRA. We integrated LETRA with TSN traffic scheduling to perform evaluations on different synthetic networks. The results show that the proposed mapping method obtains up to 90% improvement in the schedulability ratio of the traffic compared to an intuitive mapping method on a multi-switch network architecture. Through this contribution we obtain RG$_1$.

- **RC**$_3$: We propose a heuristic scheduler for TT traffic in TSN networks, called Heuristic Multi-queue Scheduler (HERMES), that takes advantage of multiple queues for TT traffic to provide high schedulability with very low scheduling times. Frames in HERMES can be configured to be scheduled in two modes of zero or relaxed reception jitter, which provides better control for users. Through a set of experiments, we show that HERMES can perform better than CP-based solutions, i.e., it results in more schedulable networks, by allowing it to use multiple queues, and at the same time, it provides the results within 17 to 800 times faster. Through this contribution we obtain our goal defined in RG$_2$.

- **RC**$_4$: 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. Then, 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. Finally, we model our proposed clock synchronization solution to verify its correctness. Through this contribution we obtain a partial solution to our goal defined in RG_3. However, through obtaining this contribution we also realize that clock integration requires specific solutions for each protocol to be integrated. This hinders adoption due to the time needed to design and implement each solution, and makes compatibility between solutions difficult. For example, if there are two legacy networks with two different synchronization protocols (P1 and P2) and we want to integrate them with TSN, we would need a specific solution for the integration of P1 with TSN and another one for P2 with TSN that might not even be compatible with each other. For this reason, we determine that it would be more efficient to palliate the problems generated by the lack of synchronization in a general way from the TSN network without involving the legacy networks.

- **RC_5**: We develop a mechanism called TALESS (TSN with Legacy End-Stations Synchronization) to address the adverse effects caused by the lack of synchronization identified through experiments on a network prototype. This solution is general and transparent for any legacy Ethernet-based network communicating through TSN. TALESS is modeled and validated through simulations with realistic network values. On the other hand, the mechanism is implemented in a network prototype to experimentally demonstrate its effectiveness. Finally, we compare the results of the experiment with the simulation model to verify both implementations. Through this contribution we obtain a complete solution to our goal defined in RG_3.

- **RC_6**: We use the **UPPAAL** model checker [14] to build a model of the SRP protocol. The model allow us to verify that SRP does not provide termination nor consistency, and to identify the scenarios in which these problems occur. We discuss the consequences derived from the absence of termination and consistency, and propose several modifications to the protocol to address these issues. Finally, we select the best one and develop a new protocol called CSRP (Consistent Stream Reservation Protocol). To ensure the correctness of our design, we create a **UPPAAL** model for CSRP and validate it through verification testing. Through this contribution we obtain our goal defined in RG_4.

6.4 Included Papers

The research contributions are proposed in the form of published papers in conferences and journals. The order of the papers is in accordance with the contributions. Five papers are included in the licentiate thesis: Paper A, B, C and E have already been published while paper D is pending for submission.

6.4.1 Paper A

Title:

LETRA: Mapping Legacy Ethernet-Based Traffic into TSN Traffic Classes

Authors:

Daniel Bujosa, Mohammad Ashjaei, Alessandro V Papadopoulos, Julian Proenza, Thomas Nolte.

Status:

Published in the 26^{th} IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2021.

Abstract:

This paper proposes a method to efficiently map the legacy Ethernet-based traffic into Time Sensitive Networking (TSN) traffic classes considering different traffic characteristics. Traffic mapping is one of the essential steps for industries to gradually move towards TSN, which in turn significantly mitigates the management complexity of industrial communication systems. In this paper, we first identify the legacy Ethernet traffic characteristics and properties. Based on the legacy traffic characteristics we presented a mapping methodology to map them into different TSN traffic classes. We implemented the mapping method as a tool, named Legacy Ethernet-based Traffic Mapping Tool or LETRA, together with a TSN traffic scheduling and performed a set of evaluations on different synthetic networks. The results show that the proposed mapping method obtains up to 90% improvement in the schedulability ratio of the traffic compared to an intuitive mapping method on a multi-switch network architecture.

Authors' Contributions:

I was the main driver of the work under the supervision of the co-authors. The plan for the paper was formed in joint discussions with the co-authors. I performed the tool implementation and evaluations and wrote the draft of the paper. The co-authors have reviewed the paper, after which I have improved it.

6.4.2 Paper B

Title:

HERMES: Heuristic Multi-queue Scheduler for TSN Time-Triggered Traffic with Zero Reception Jitter Capabilities

Authors:

Daniel Bujosa, Mohammad Ashjaei, Alessandro V Papadopoulos, Julian Proenza, Thomas Nolte.

Status:

Published in the 30^{th} International Conference on Real-Time Networks and Systems (RTNS), 2022.

Abstract:

The Time-Sensitive Networking (TSN) standards provide a toolbox of features to be utilized in various application domains.The core TSN features include deterministic zero-jitter and low-latency data transmission and transmitting traffic with various levels of time-criticality on the same network. To achieve a deterministic transmission, the TSN standards define a time-aware shaper that coordinates transmission of Time-Triggered (TT) traffic. In this paper, we tackle the challenge of scheduling the TT traffic and we propose a heuristic algorithm, called HERMES. Unlike the existing scheduling solutions, HERMES results in a significantly faster algorithm run-time and a high number of schedulable networks. HERMES can be configured in two modes of zero or relaxed reception jitter while using multiple TT queues to improve the schedulability. We compare HERMES with a constraint programming (CP)-based solution and we show that HERMES performs better than the CP-based solution if multiple TT queues are used, both with respect to algorithm run-time and schedulability of the networks.

Authors' Contributions:

I was the main driver of the work under the supervision of the co-authors. The plan for the paper was formed in joint discussions with the co-authors. I performed the tool implementation and evaluations and wrote the draft of the paper. The co-authors have reviewed the paper, after which I have improved it.

6.4.3 Paper C

Title:

Clock Synchronization in Integrated TSN-EtherCAT Networks

Authors:

Daniel Bujosa, Daniel Hallmans, Mohammad Ashjaei, Alessandro V Papadopoulos, Julian Proenza, Thomas Nolte.

Status:

Published in the 25^{th} IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2020.

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.

Authors' Contributions:

I and Daniel Hallmans were the main drivers of the work under the supervision of the co-authors. The plan for the paper was formed in joint discussions with the co-authors. I performed the tool implementation and evaluations, and I wrote the draft of the paper in collaboration with Daniel Hallmans. The co-authors reviewed the paper, after which I improved it.

6.4.4 Paper D

Title:

TALESS: TSN with Legacy End-Stations Synchronization

Authors:

Daniel Bujosa, Mohammad Ashjaei, Alessandro V Papadopoulos, Julian Proenza, Thomas Nolte.

Status:

Pending for submission to the IEEE Transactions on Industrial Informatics.

Abstract:

Time Sensitive Networks (TSN) have become one of the most important communications standards in many industrial sectors. However, TSN requires specific hardware and software. This makes it difficult for established companies to adopt TSN, as it would imply changing most legacy hardware and software which may not be cost-effective in most cases. In order to enable the adoption of TSN by the industry and be more environmentally sustainable, it is necessary to develop tools to integrate legacy systems with TSN. In this paper, we propose a solution for the coexistence of different time domains from different legacy subsystems with their corresponding synchronization protocols in a single TSN network. To this end, we experimentally identified the effects of replacing the communications subsystem of a legacy Ethernet-based network with TSN in terms of synchronization. Based on the results, we propose a solution called TSN with Legacy End-Stations Synchronization (TALESS). TALESS is able to identify the drift between the TSN communications subsystem and the legacy devices and modify the TSN schedule to adapt to the time domains of the integrated legacy systems in order to avoid the effects of the lack of synchronization between them. We validate TALESS through both simulations and experiments on a prototype. Thereby we demonstrate that, thanks to TALESS, legacy systems are able to synchronize through TSN and even improve features such as their reception jitter or their integrability with other legacy systems.

Authors' Contributions:

I was the main driver of the work under the supervision of the co-authors. The plan for the paper was formed in joint discussions with the co-authors. I performed the tool implementation and evaluations and wrote the draft of the paper. The co-authors have reviewed the paper, after which I have improved it.

6.4.5 Paper E

Title:

CSRP: An Enhanced Protocol for Consistent Reservation of Resources in AVB/TSN

Authors:

Daniel Bujosa, Ines Álvarez, Julian Proenza.

Status:

Published in IEEE Transactions on Industrial Informatics, 2020.

Abstract:

The IEEE Audio Video Bridging (AVB) Task Group (TG) was created to provide Ethernet with soft real-time guarantees. Later on, the TG was renamed to Time-Sensitive Networking (TSN) and its scope broadened to support hard real-time and critical applications. The Stream Reservation Protocol (SRP) is a key work of the TGs as it allows reserving resources in the network, guaranteeing the required quality of service (QoS). AVB's SRP is based on a distributed architecture, while TSN's is based on centralized ones. The distributed version of SRP is supported and used in TSN. Nevertheless, it was not designed to provide properties that are important for critical applications. In this work we model SRP using UPPAAL and we study the termination and consistency. We verify that SRP does not provide such properties. Furthermore, we propose an improved protocol called Consistent Stream Reservation Protocol (CSRP) and we formally verify its correctness using UPPAAL.

Authors' Contributions:

I was the main driver of the work under the supervision of the co-authors. The plan for the paper was formed in joint discussions with the co-authors. I performed the tool implementation and evaluations and wrote the draft of the paper. The co-authors have reviewed the paper, after which I have improved it.

6.4.6 Mapping the Included Papers with the Research Contributions

The mapping of the aforementioned thesis contribution into published and planned publications, that are included in the thesis, is shown in Table 6.1.

Given the overall set of contributions towards the research goals of this thesis we believe that in summary the thesis make a significant step towards achieving the overall goal of the thesis.

Table 6.1: The mapping of the research goals to the papers included in the thesis.

	Paper A	**Paper B**	**Paper C**	**Paper D**	**Paper E**
RC$_1$	✓				
RC$_2$	✓				
RC$_3$		✓			
RC$_4$			✓		
RC$_5$				✓	
RC$_6$					✓

Chapter 7

Summary and Future Work

7.1 Summary

In this thesis we have developed tools and mechanisms for the migration and integration of legacy networks into TSN as well as improved some of its mechanisms. In this way, we seek to facilitate the adoption of TSN avoiding the time, effort, and resources that would be required to replace a functional network with a new TSN network. Furthermore, thanks to the proposed solutions, legacy networks can not only communicate through TSN normally, which improves their integration with other legacy networks and reduces the overall complexity thanks to the use of a single communication protocol, but they are also able to benefit from the TSN enhancements providing a better service.

We have improved TSN's SRP to avoid wasting resources (RC_4) and we have created three tools aimed at migrating and integrating legacy networks with TSN. These tools include: firstly, LETRA, a tool for mapping legacy traffic into the different types of TSN traffic (RC_1); secondly, HERMES, which is a heuristic scheduling algorithm implemented as a fast and scalable tool capable of scheduling the traffic mapped by LETRA (RC_2); and finally, TALESS, a synchronization tool capable of eliminating the adverse effects caused by the lack of synchronization between legacy networks and TSN (RC_3).

7.2 Future Work

As future work, we seek to develop a methodology for traffic identification in legacy networks. This will be able to, by sampling traffic from an existing network or by analyzing the network specification, parameterize the legacy traffic in terms of relevant parameters for the migration of such traffic to TSN.

In addition, we will develop a TSN frame forging mechanisms for Ethernet frames. It will be able to modify the Ethernet frame fields of legacy traffic dynamically to conform to the TSN format, e.g., by assigning TSN priorities. This will enable the transmission of legacy traffic as a specific TSN traffic type transparently to the legacy end-stations.

Finally, these tools will be integrated with those already presented in this work. This will provide a set of tools capable of identifying traffic, mapping it, scheduling it, and synchronizing it all in one. Such set of tools would considerably reduce migration and integration times and allow adoption of TSN with minimal costs.

Bibliography

[1] IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams. *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pages C1–72, 2010.

[2] IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP). *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*, 2010.

[3] IEEE Standard for Local and Metropolitan Area Networks–Audio Video Bridging (AVB) Systems. *IEEE Std 802.1BA-2011*, pages 1–45, 2011.

[4] IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks – Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements. *IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018)*, pages 1–208, Oct 2018.

[5] IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications. *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pages 1–421, 2020.

[6] 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.

[7] Inés Álvarez, Ignasi Furió, Julián Proenza, and Manuel Barranco. Design and Experimental Evaluation of the Proactive Transmission of Replicated Frames Mechanism over Time-Sensitive Networking. *Sensors*, 21(3):756, 2021.

[8] 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.

[9] Emmanuel Arzuaga and David R Kaeli. Quantifying load imbalance on virtualized enterprise servers. In *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, pages 235–242, 2010.

[10] Mohammad Ashjaei, Lucia Lo Bello, Masoud Daneshtalab, Gaetano Patti, Sergio Saponara, and Saad Mubeen. Time-Sensitive Networking in Automotive Embedded Systems: State of the Art and Research Opportunities. *Journal of Systems Architecture*, 110:1–47, September 2021.

[11] Mohammad Ashjaei, Gaetano Patti, Moris Behnam, Thomas Nolte, Giuliana Alderisi, and Lucia Lo Bello. Schedulability analysis of Ethernet Audio Video Bridging networks with scheduled traffic support. *Real-Time Systems*, 53(4):526–577, 2017.

[12] Ayman A Atallah, Ghaith Bany Hamad, and Otmane Ait Mohamed. Fault-resilient topology planning and traffic configuration for IEEE 802.1 Qbv TSN networks. In *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, pages 151–156. IEEE, 2018.

[13] Haytham Baniabdelghany, Roman Obermaisser, et al. Extended synchronization protocol based on IEEE802. 1AS for improved precision in dynamic and asymmetric TSN hybrid networks. In *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–8. IEEE, 2020.

[14] Gerd Behrmann, Alexandre David, and Kim G. Larsen. *A Tutorial on Uppaal*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[15] D. Bujosa, I. Alvarez, D. Čavka, and J. Proenza. Analysing Termination and Consistency in the AVB's Stream Reservation Protocol. In *Proceedings of the IEEE 24th International Conference on Emerging Technologies and Factory Automation (ETFA 2019)*, October 2019.

[16] Daniel Bujosa, Andreas Johansson, Mohammad Ashjaei, Alessandro V Papadopoulos, Julian Proenza, and Thomas Nolte. The Effects of Clock Synchronization in TSN Networks with Legacy End-Stations. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4. IEEE, 2022.

[17] Zichao Chai, Wei Liu, Mao Li, and Jing Lei. Cross Domain Clock Synchronization Based on Data Packet Relay in 5G-TSN Integrated Network. In *2021 IEEE 4th International Conference on Electronics and Communication Engineering (ICECE)*, pages 141–145. IEEE, 2021.

[18] Sameer Chouksey, Hariram Selvamurugan Satheesh, and Johan Åkerberg. An experimental study of TSN-nonTSN coexistence. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 1577–1584. IEEE, 2021.

[19] Industrial Internet Consortium et al. Time sensitive networks for flexible manufacturing testbed characterization and mapping of converged traffic types, 2019.

[20] Silviu S Craciunas, Ramon Serna Oliver, Martin Chmelík, and Wilfried Steiner. Scheduling Real-Time Communication in IEEE 802.1 Qbv Time Sensitive Networks. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, pages 183–192, 2016.

[21] Théo Docquier, Ye-Qiong Song, Vincent Chevrier, Ludovic Pontnau, and Abdelaziz Ahmed-Nacer. Iec 61850 over tsn: Traffic mapping and delay analysis of goose traffic. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 246–253. IEEE, 2020.

[22] Gordana Dodig-Crnkovic. Scientific Methods in Computer Science. In *Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges*, 2002.

[23] Voica Gavriluţ and Paul Pop. Traffic-Type Assignment for TSN-Based Mixed-Criticality Cyber-Physical Systems. *ACM Trans. Cyber-Phys. Syst.*, 4(2), 2020.

[24] Voica Gavriluţ and Paul Pop. Scheduling in Time Sensitive Networks (TSN) for mixed-criticality industrial applications. In *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, pages 1–4, 2018.

[25] Voica Gavriluţ, Luxi Zhao, Michael L Raagaard, and Paul Pop. AVB-aware Routing and Scheduling of Time-Triggered traffic for TSN. *IEEE Access*, 6:75229–75243, 2018.

[26] Michael Gundall, Christopher Huber, Peter Rost, Rüdiger Halfmann, and Hans D Schotten. Integration of 5G with TSN as prerequisite for a highly flexible future industrial automation: Time synchronization based on IEEE 802.1 AS. In *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, pages 3823–3830. IEEE, 2020.

[27] Marina Gutiérrez, Astrit Ademaj, Wilfried Steiner, Radu Dobrin, and Sasikumar Punnekkat. Self-configuration of IEEE 802.1 TSN networks. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2017.

[28] Marina Gutiérrez, Wilfried Steiner, Radu Dobrin, and Sasikumar Punnekkat. A Configuration Agent based on the Time-Triggered paradigm for Real-Time Networks. In *2015 IEEE World Conference on Factory Communication Systems (WFCS)*, pages 1–4. IEEE, 2015.

[29] Jetmir Haxhibeqiri, Xianjun Jiao, Muhammad Aslam, Ingrid Moerman, and Jeroen Hoebeke. Enabling TSN over IEEE 802.11: Low-overhead time synchronization for wi-fi clients. In *2021 22nd IEEE international conference on industrial technology (ICIT)*, volume 1, pages 1068–1073. IEEE, 2021.

[30] IEEE. Ieee standard for local and metropolitan area networks–bridges and bridged networks, 2018. Includes TSN Time-Aware Shaper (TAS) definition, among other TSN-related features and protocols.

[31] 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.

[32] O. Kleineberg, P. Fröhlich, and D. Heffernan. Fault-Tolerant Ethernet Networks with Audio and Video Bridging. In *ETFA2011*, pages 1–8, Sept 2011.

[33] Leonard Kleinrock. Queueing systems, volume i: Theory, vol. I, 1975.

[34] H. Lim, L. Völker, and D. Herrscher. Challenges in a Future IP/Ethernet-based in-car Network for Real-Time Applications. In *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 7–12, June 2011.

[35] Lucia Lo Bello, Mohammad Ashjaei, Gaetano Patti, and Moris Behnam. Schedulability analysis of Time-Sensitive Networks with scheduled traffic and preemption support. *Journal of Parallel and Distributed Computing,*, 144,, 2020.

[36] P. Meyer, T. Steinbach, F. Korf, and T. C. Schmidt. Extending IEEE 802.1 AVB with Time-Triggered Scheduling: A Simulation Study of the Coexistence of Synchronous and Asynchronous Traffic. In *2013 IEEE Vehicular Networking Conference*, pages 47–54, Dec 2013.

[37] Jörn Migge, Josetxo Villanueva, Nicolas Navet, and Marc Boyer. Insights on the Performance and Configuration of AVB and TSN in Automotive Ethernet Networks. In *9th European Congress on Embedded Real Time Software and Systems (ERTS 2018)*, Toulouse, France, January 2018.

[38] Seifeddine Nsaibi, Ludwig Leurs, and Hans D Schotten. Formal and simulation-based timing analysis of Industrial-Ethernet sercos III over TSN. In *2017 IEEE/ACM 21st International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–8, 2017.

[39] Maryam Pahlevan and Roman Obermaisser. Genetic Algorithm for scheduling Time-Triggered traffic in Time-Sensitive Networks. In *2018 IEEE 23rd international conference on emerging technologies and factory automation (ETFA)*, volume 1, pages 337–344. IEEE, 2018.

[40] Maryam Pahlevan, Nadra Tabassam, and Roman Obermaisser. Heuristic list scheduler for Time Triggered traffic in Time Sensitive Networks. *ACM Sigbed Review*, 16(1):15–20, 2019.

[41] Rina Panigrahy, Kunal Talwar, Lincoln Uyeda, and Udi Wieder. Heuristics for vector bin packing. *research.microsoft.com*, 2011.

[42] D. Park, J. Lee, C. Park, and S. Park. New Automatic De-Registration Method Utilizing a Timer in the IEEE802.1 TSN. In *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, pages 47–51, Oct 2016.

[43] R. Queck. Analysis of Ethernet AVB for Automotive Networks using Network Calculus. In *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*, pages 61–67, July 2012.

[44] Michael Lander Raagaard and Paul Pop. Optimization algorithms for the scheduling of IEEE 802.1 Time-Sensitive Networking (TSN). *Tech. Univ. Denmark, Lyngby, Denmark, Tech. Rep*, 2017.

[45] Mauricio GC Resende and Celso C Ribeiro. GRASP: Greedy Randomized Adaptive Search Procedures. In *Search methodologies*, pages 287–312. Springer, 2014.

[46] Alexey M Romanov, Francesco Gringoli, and Axel Sikora. A precise synchronization method for future wireless TSN networks. *IEEE Transactions on Industrial Informatics*, 17(5):3682–3692, 2020.

[47] Siwar Ben Hadj Said, Quang Huy Truong, and Michael Boc. SDN-based configuration solution for IEEE 802.1 Time Sensitive Networking (TSN). *ACM SIGBED Review*, 16(1):27–32, 2019.

[48] R. Salazar, T. Godfrey, L. Winkel, N. Finn, C. Powell, B. Rolfe, and M. Seewald. Utility Applications of Time Sensitive Networking White Paper (D3). Technical report, IEEE, 2018.

[49] S. Samii and H. Zinner. Level 5 by Layer 2: Time-Sensitive Networking for Autonomous Vehicles. *IEEE Communications Standards Magazine*, 2(2):62–68, 2018.

[50] Eike Schweissguth, Dirk Timmermann, Helge Parzyjegla, Peter Danielis, and Gero Mühl. ILP-based routing and scheduling of multicast realtime traffic in Time-Sensitive Networks. In *2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–11. IEEE, 2020.

[51] Oscar Seijo, Zaloa Fernández, Iñaki Val, and Jesús A López-Fernández. SHARP: towards the integration of Time-Sensitive communications in legacy LAN/WLAN. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–7, 2018.

[52] Oscar Seijo, Xabier Iturbe, and Inaki Val. Tackling the Challenges of the Integration of Wired and Wireless TSN with a Technology Proof-of-Concept. *IEEE Transactions on Industrial Informatics*, 2021.

[53] Haochuan Shi, Adnan Aijaz, and Nan Jiang. Evaluating the Performance of Over-the-Air Time Synchronization for 5G and TSN Integration. In *2021 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pages 1–6. IEEE, 2021.

[54] Jiajia Song, Makoto Kubomi, Jeffrey Zhao, and Daisuke Takita. Time synchronization performance analysis considering the frequency offset inside 5G-TSN network. In *2021 17th International Symposium on Wireless Communication Systems (ISWCS)*, pages 1–6. IEEE, 2021.

[55] Tobias Striffler and Hans D Schotten. The 5G Transparent Clock: Synchronization Errors in Integrated 5G-TSN Industrial Networks. In *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, pages 1–6. IEEE, 2021.

[56] Thomas Stüber, Lukas Osswald, Steffen Lindner, and Michael Menth. A Survey of Scheduling in Time-Sensitive Networking (TSN). *arXiv preprint arXiv:2211.10954*, 2022.

[57] Ammad Ali Syed, Serkan Ayaz, Tim Leinmüller, and Madhu Chandra. Dynamic Scheduling and Routing for TSN based in-vehicle networks. In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2021.

[58] K. S. Umadevi and R. K. Sridharan. Multilevel ingress scheduling policy for Time Sensitive Networks. In *Int. Conf. Microelectronic Devices, Circuits and Systems*, 2017.

[59] M. Wollschlaeger, T. Sauter, and J. Jasperneite. The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0. *IEEE Industrial Electronics Magazine*, 11(1):17–27, 2017.

[60] Jin Y Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971.

[61] Luxi Zhao, Paul Pop, Zhong Zheng, and Qiao Li. Timing analysis of AVB traffic in TSN networks using network calculus. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 25–36, 2018.

II

Included Papers

Chapter 8

Paper A
LETRA: Mapping Legacy Ethernet-Based Traffic into TSN Traffic Classes.

Daniel Bujosa, Mohammad Ashjaei, Alessandro V. Papadopoulos, Julián Proenza, Thomas Nolte.
In the 26^{th} IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2021).

Abstract

This paper proposes a method to efficiently map the legacy Ethernet-based traffic into Time Sensitive Networking (TSN) traffic classes considering different traffic characteristics. Traffic mapping is one of the essential steps for industries to gradually move towards TSN, which in turn significantly mitigates the management complexity of industrial communication systems. In this paper, we first identify the legacy Ethernet traffic characteristics and properties. Based on the legacy traffic characteristics we presented a mapping methodology to map them into different TSN traffic classes. We implemented the mapping method as a tool, named Legacy Ethernet-based Traffic Mapping Tool or LETRA, together with a TSN traffic scheduling and performed a set of evaluations on different synthetic networks. The results show that the proposed mapping method obtains up to 90% improvement in the schedulability ratio of the traffic compared to an intuitive mapping method on a multi-switch network architecture.

8.1 Introduction

New technologies often offer new solutions or improvements for companies that can lead to an advantage over the competitors; reducing costs or offering a better product, or lead to an environmental improvement; improving performance, optimization of resources, and/or emission reduction. However, new opportunities imply new challenges. These challenges are often related to their integration with existing legacy solutions and their implementation. For many industrial applications, it may not be cost-effective to adopt the new technologies, as it may require redeveloping all previous solutions and systems.

One of the new technologies, which can change the current paradigm of industrial communications and seems to be a key to the transition to Industry 4.0, is Time-Sensitive Networking (TSN). Everything started when, in 2005, the IEEE Audio-Video Bridging (AVB) Task Group (TG) was created. The purpose was to provide Ethernet with soft real-time capabilities oriented to audio/video streaming. The AVB TG developed three projects: (i) the IEEE Std 802.1AS [5], dedicated to clock synchronization, (ii) the IEEE Std 802.1Qav, which standardized the Credit-Based Shaper (CBS) [1]; and, finally, the IEEE Std 802.1Qat, which standardized the Stream Reservation Protocol (SRP) [2]. Additionally, another profile with series of rules, called IEEE Std 802.1BA-2011: Audio Video Bridging Systems [3], was created to ensure a minimum QoS when using the aforementioned standards. These standards together are commonly referred to as AVB standards. Over time, areas of applications, such as automotive [18], automation [21] and energy distribution [17], were interested in the work done by the TG so in 2012 the group was renamed to TSN TG and its objective was expanded to meet the needs of these new applications. The set of standards developed by the TG is usually referred to as TSN standards and it presents several interesting features. Specifically, TSN seems to provide Ethernet with proper support for mixed hard and soft real-time communications, flexibility of the traffic requirements and fault tolerance mechanisms. For these characteristics, TSN seems promising to enable new solutions within the context of modern industrial systems and solutions enabling, among other things, the integration of multiple legacy networks onto one TSN network. However, current TSN networks do not support all Ethernet-based legacy system message implementation characteristics, such as jitter in some legacy network devices, while currently used legacy technologies do not meet all TSN requirements. Moreover, it is cost-effective and beneficial for companies if they gradually move towards new technologies instead of completely replacing existing ones. Therefore, solutions to integrate a legacy system into a TSN network are essential so that services are not disturbed. An example of

that is when a network consists of 4 nodes connected via different protocols, would be able to replace the end-to-end links with a TSN switch connected to all 4 nodes. This would allow the communications between networks while letting previous communications benefit from TSN features hence improving their real-time behavior, synchronization, and fault tolerance, to name a few.

Contributions: To allow industry to adopt TSN solutions, and the desire integration, a proper migration methodology of the legacy Ethernet-based traffic to TSN traffic classes should be designed. In this work we propose three steps to achieve this goal, as follows:

1. We develop a Legacy Ethernet-based Traffic model that can describe messages of any Ethernet-based communication protocols. Moreover, apart from the model, we require a methodology to identify the parameters of the messages inherited from the legacy system. The identification methodology is out of the scope of this paper. However, some works like [12] have already addressed this issue.

2. We develop a mapping methodology, and its corresponding implementation as a mapping tool, named Legacy Ethernet-based Traffic Mapping Tool or LETRA, that can map the legacy Ethernet-based messages characterized by the proposed model into different TSN traffic classes. To the best of our knowledge, this is the first attempt to map the Ethernet-based legacy traffic into TSN traffic classes considering a full spectrum of message characteristics. To evaluate the mapping methodology we implemented it as a mapping tool and compared its performance with an intuitive mapping methodology on different networks.

3. We integrated a pre-existing TSN scheduling method and a TSN schedulability analysis method into LETRA, which can map the messages, schedule the TSN traffic and evaluate their real-time behavior.

Paper outline: The paper is organized as follows. Section 8.2 presents the related work. Section 8.3 presents the legacy Ethernet-based traffic model. Section 8.4 describes the background of TSN and TSN traffic classes. Then, Section 8.5 proposes the traffic mapping methodology and development of LETRA, while Section 8.6 presents the experiments and evaluations. Finally, Section 8.7 concludes the paper and gives future directions.

8.2 Related work

Due to the great relevance of the work done by the TSN TG since 2012, the community has carried out a significant amount of work related to their study, application, and improvement. For example, the work in [6] studied the effects of the time-aware shaper, the work in [13] analyzed the fault tolerance issues, while the work in [7] proposed time redundancy to tolerate temporary faults, the work in [20] studied the scheduling policies and the load balancing was studied in [8]. Moreover, the work in [9] provided an up-to-date comprehensive survey of the TSN-related research.

Within the context of TSN traffic mapping, the work in [12] presented a network monitoring method to obtain the traffic properties based on measurements. A meta-heuristic method is proposed in [11] that maps mixed-criticality applications into the TSN traffic classes. Although the aim is similar to this paper, the proposed method does not cover all cases that are studied and exist in industrial applications. The method, thus, becomes suitable for cases where only very few mixed-criticality levels are assumed in the legacy system with no extensive timing information, whereas in this paper we consider different traffic characteristics including many timing characteristics and constraints when we map them into the TSN traffic classes.

There are also few works on integrating legacy networks into TSN networks. For example, the work in [15] integrated a few of the TSN standards into Sercos III, which is a closed system that allows standard Ethernet devices to be plugged, to improve its performance. Moreover, an integration methodology of wireless TSN (802.11) was proposed in [19]. Finally, an integration that focuses on the clock synchronization for EtherCAT and TSN was proposed in [14].

Nevertheless, to be the best of our knowledge, the proposed mapping methodology and the tool LETRA in this paper is the first attempt to map messages from any Ethernet-based legacy network into TSN traffic classes considering a full spectrum of message characteristics.

8.3 Legacy Ethernet-based traffic model

This section introduces a message model to describe the legacy Ethernet-based messages. To handle legacy messages from different Ethernet-based protocols, we identified a set of characteristics with which we can model the Ethernet-based messages. The model is used by the mapping tool to map the messages into different TSN traffic classes. Note that the extraction of values in the model is out of the scope of this paper, which can be done by measurements in

the legacy networks as described in [12].

The parameters used to characterize the Ethernet messages are divided into three categories, including: *common parameters*, *periodic message parameters* and *non-periodic message parameters*. Note that not all messages need to have all parameters to map them. Following is the description of the parameters.

8.3.0.1 Common parameters

The common parameters are those that are independent of the behavior of the messages that are presented by $\{S, D, ML, DL, LRT, FLR, Prec\}$. In the above set, S and D represent the source and destination(s) of the message. The length of the message in bytes is denoted by ML which can be a range of sizes to consider variable message sizes. The message deadline is shown by DL. Moreover, LRT shows if the message is soft or hard real-time, i.e., missing deadlines lead to a degradation of functionality or a complete system failure, respectively. FLR denotes the message lost percentage, and $Prec$ denotes the precedence constraints between two or more messages which identifies if some messages should be transmitted or received in a specific sequential order.

8.3.0.2 Periodic message parameters

The periodic message parameters are those dependent on the periodicity of the messages, that is presented by $\{P, O, JI, JO\}$. P are the message period, while O is the offset, i.e., time shift of the message release time. JI represents the maximum jitter on the message release, while JO is the maximum jitter in the message reception. Note that jitter is the variation of delays that can be on transmission and/or reception of the message.

8.3.0.3 Non-periodic message parameters

The main parameter for non-periodic messages is the minimum inter-arrival time, identified by MIT, which is the minimum time between two consecutive releases of a message in non-periodic messages.

Figure 8.1 illustrates a graphical representation of some of the presented parameters in the model.

8.4 TSN traffic characteristics

This section gives a brief background about the TSN standards and TSN traffic classes. Communication in a TSN network is done among end-stations through

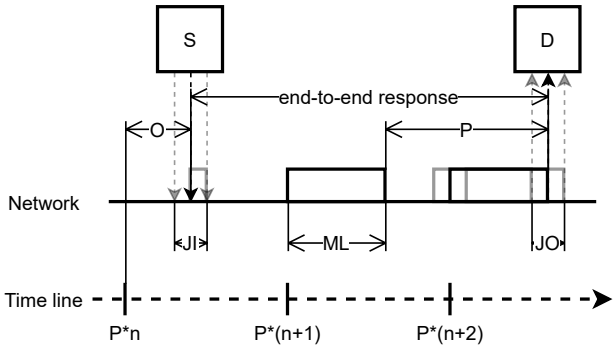


Figure 8.1: Graphical representation of the presented model.

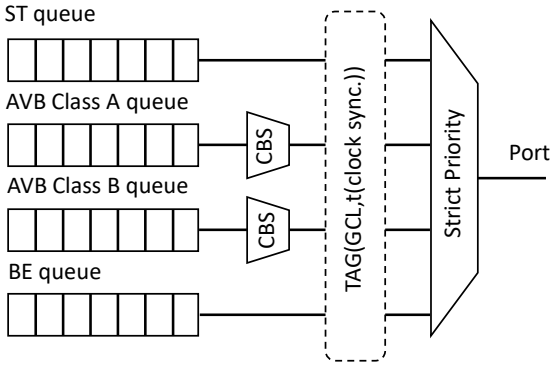


Figure 8.2: A TSN egress port.

routes of links and switches through Ethernet messages. A port in a TSN switch supports eight FIFO queues. A typical port with four TSN traffic classes (queues) is shown in Fig. 8.2. The TSN standards define three traffic classes including Scheduled Traffic (ST), Audio-Video Bridging (AVB) traffic, and Best Effort (BE) traffic. The AVB traffic is named by classes A and B, where class A has higher priority than class B. Following, we describe the details of TSN traffic classes.

8.4.1 ST traffic class

The ST traffic is scheduled offline, which makes them fully deterministic with zero jitter in the message delivery. The TSN standard [4] defines a Time-Aware Gates (TAG) that is controlled by a Gate Control List (GCL). The GCL specifies at which specific time of the network-wide reference time gates are

open, and thus, the link is available for a queue to send messages. Note that the reference clock is achieved thanks to the synchronization protocol defined in [5] which allows clock synchronization between end-stations and switches.

8.4.2 AVB traffic class

The AVB TG [1] introduced the CBS that defines credits to AVB queues. The credit is consumed when a message in that queue is sent, otherwise, it is replenished when there is a pending message in the queue (or the credit is still negative). The AVB queues can only transmit when their credit is positive or zero and their gate is open according to the GCL. CBS defines priority classes (classes A and B) but allowing transmission of low-priority traffic even if high-priority traffic is waiting according to their credits. This reduces buffering and improves lower-priority traffic QoS. Even though the activation time of AVB traffic is unknown due to possible blocking from other AVB classes or ST queues, there are analysis methods to calculate their worst-case response time. For example, the analysis used in the experiments of this paper is the one presented in [10].

8.4.3 BE traffic class

BE has no real-time guarantees and is the lowest priority. This queue is not shaped by CBS and can only be sent if its gate is open and all other AVB queues have negative credit or there is no AVB traffic ready for transmission.

8.5 Proposed traffic mapping methodology

To map the legacy Ethernet-based traffic characterized by the parameters modeled in Section 8.3 into the TSN traffic classes, we developed three logic-based equations explained in the following sub-sections. As the equations are logic-based, in this section all parameters are treated as logical Boolean variables. In this sense, a parameter is True if the frame is affected by the parameter, otherwise False. For example, if one frame is non-periodic and has deadline of $100ms$, as $P = NULL$, $P = 0$ in the equation. On the other hand, as $DL = 100ms$, $DL = 1$ in the equation.

8.5.1 Mapping to the ST traffic class

The mapping first checks whether the legacy Ethernet traffic should be mapped into the ST traffic, according to the following expression:

$$ST = P \,\&\, (JO \parallel (!JI \,\&\, DL)) \tag{8.1}$$

where $\&$ is the logical "and" operator, $\parallel$ is the logical "or" operator, and ! is the logical "not" operator. First of all, Eq. (8.1) checks whether the message is periodic (P). This is mandatory as, otherwise, it would be impossible to implement a proper GCL. That is because, as described in Section 8.4, GCL is a list of open/close instructions executed repeatedly at certain times. Therefore, even if it is possible to schedule the message instances offline it would imply generating a long GCL table that makes the schedule impractical. Secondly, Eq. (8.1) checks if the message has JO constraints, and it is periodic, it must be transmitted as ST traffic as it is the only way to ensure meeting the JO requirement. That is because, as mentioned in Section 8.4, ST traffic is the only fully deterministic TSN traffic with zero JO. However, if the message has no JO requirements but it has DL and no JI then it can be also sent as an ST message. The reason for having D is to benefit from the ST characteristics while the JI conditions are to prevent waste of resources. JI implies considering larger open gates for the ST traffic to ensure the message instance to be transmitted within that time. This intuitively means allocating more bandwidth, which can be a waste of network bandwidth resources. Note that this happens only with JI and not with JO as once the message reaches the first TSN switch the messages will be sent just when the window starts.

8.5.2 Mapping to the AVB traffic class

Eq. (8.2) indicates whether the message can be sent as an AVB message. First, the message should have DL constraints to benefit from being sent as an AVB message. Moreover, the message must not have JO constraints unless it is not a hard real-time message. If the message has JO constraints but it is not hard real-time, another type of analysis, such as utilization-based analysis [11] may be needed. Note that Eq. (8.2) only specifies if the message can be transmitted as AVB traffic but it says nothing about the AVB possible priority classes. In this work, we consider only one AVB class queue for mapping as currently LETRA only identifies the messages as suitable or not for each traffic class. Later, it will be integrated or improved through constraint programming and/or metaheuristics to reach the desired specification level. In Eq. (8.2), HRT indicates the hard real-time requirement for the message.

$$AVB = DL \,\&\, (!JO \parallel !HRT) = DL \,\&\, !(JO \,\&\, HRT) \tag{8.2}$$

8.5.3 Mapping to the BE traffic class

Eq. (8.3) indicates whether the message can be sent as a BE class message. It checks whether the message has real-time requirements or not, which is the only requirement to be in the BE class.

$$BE = !JO \,\&\, !DL = !(JO \parallel DL) \tag{8.3}$$

8.5.4 Resulting truth table

As a result of the presented mapping methodology, we summarized the equations with a truth table shown in Table 8.1. As it can be seen, we just use P, JI, JO, DL, and LRT to map legacy Ethernet messages. The other parameters in the model do not affect the mapping but on the scheduling and analysis of the traffic after the mapping is performed to verify the timing properties.

To show the performance of the proposed mapping methodology we also define an intuitive mapping methodology. The intuitive mapping methodology classifies all periodic messages as ST traffic class and all non-periodic messages as AVB traffic class to still have a level of timing guarantee for them.

Note that the presented mapping methodology cannot be compared with the mapping presented in [11], which is only based on the criticality level of messages. The reason is that we consider more specific variables to map the messages, which means that [11] cannot map 90% of the messages considered in this work. However, [11] maps messages between ST class and AVB class which, according to our tool, are suitable for both classes. In this sense, the combination of both tools would expand the number of mappable messages, while resolving some ambiguities in LETRA.

8.5.5 Evaluation tool

To have a complete evaluation tool, we implemented LETRA, a TSN traffic scheduling, and a schedulability analysis to be able to evaluate the solution. Moreover, we developed a network generator and an intuitive mapping tool which implements the intuitive mapping methodology described in Section 8.5. The tools that are used for this evaluation from previous works include the ST scheduling tool in [16] and the AVB traffic schedulability analysis in [10].

P	JI	JO	DL	HRT	ST	AVB	BE
0	X	X	0	0	0	0	1
0	X	X	0	1	0	0	1
0	X	X	1	0	0	1	0
0	X	X	1	1	0	1	0
1	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1
1	0	0	1	0	1	1	0
1	0	0	1	1	1	1	0
1	0	1	0	0	1	0	0
1	0	1	0	1	1	0	0
1	0	1	1	0	1	1	0
1	0	1	1	1	1	0	0
1	1	0	0	0	0	0	1
1	1	0	0	1	0	0	1
1	1	0	1	0	0	1	0
1	1	0	1	1	0	1	0
1	1	1	0	0	1	0	0
1	1	1	0	1	1	0	0
1	1	1	1	0	1	1	0
1	1	1	1	1	1	0	0

Table 8.1: Truth table of the mapping methodology.

The integration of the mentioned tools is shown in Fig. 8.3. First, the Network Generator generates the network messages according to the network configuration specified in the presented model. The generated messages are used as inputs for LETRA and the intuitive mapping tool, thus, obtaining two different classifications for the messages. The ST messages of both mapping tools are scheduled through the ST traffic scheduler [16] and, finally, the AVB traffic is checked through the AVB analyzer [10], which checks whether the AVB traffic are schedulable considering the transmission algorithms based on a response time analysis. The results of the scheduler and the AVB analysis are used to compare the mapping performance.

The Network Generator tool uses the parameters in the model explained in Section 8.3 as inputs to generate the messages randomly. Besides the parameters in the model, the network topology is an input.

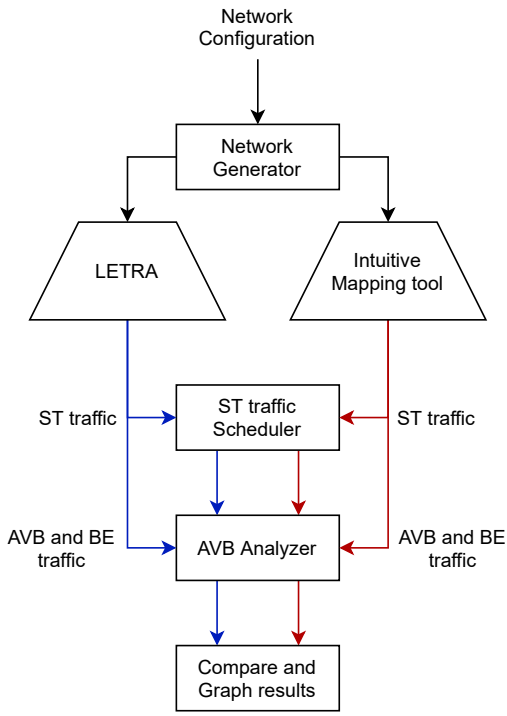


Figure 8.3: Integration of the tools for evaluation.

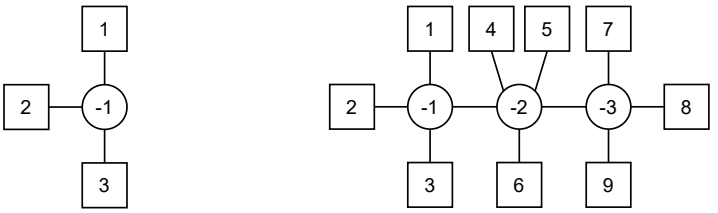
8.6 Experiments and results

This section presents the experiments that we conducted to evaluate the proposed mapping methodology using the developed integrated tools (Fig. 8.3). We first present the experimental setup and then we illustrate and discuss the results.

8.6.1 Experimental setup

For the evaluation in this paper, we considered two network architectures, including a single-switch and a three-switch network. The topology is a line-star topology with the switches connected in a line and nodes connected to the switches in the form of stars, as shown in Fig. 8.4. The line topology, apart from being widely used in the industry in many layers of the automation pyramid, is simpler and share many similarities with a tree topology. This allows us to extend the results of these experiments to a greater number of communication networks that could be developed in the future.

The network generator is designed such that we can select the input prob-



(a) A single-switch network. **(b)** A three-switches network.

Figure 8.4: Experimental network architectures.

abilities to uniformly distribute the probability among all possible messages, which is listed in Table 8.1. To achieve that we set the probability of all parameters to 50% except for periodicity. This means, as an example, that the messages have a 50% chance to have deadline constraints. With this setup, we ensure that all possible combinations with the same probability will be generated.

The parameters to generate the messages are set as follows. The network bandwidth is set to 10Mbps to prevent generating too many messages in case of generating a high load. The maximum link utilization is varied within the range [10%,90%] with the interval of 5%. Note that the messages will be generated such that the utilization in all links will be the one selected as an input, i.e., when we select 10% utilization the message generator selects the message sizes and routes to obtain 10% on all links. The maximum number of generated messages is set to 100, however, depending on the selected link utilization the message number can be different. The message length is selected within the range [64,1530] Bytes. The maximum allowed period and minimum inter-arrival time for messages are set to 1000μs. We also allowed arbitrary deadlines, which can be selected within the range [500,1000]μs. The input and output jitter values are also selected within the range [1,100]μs. We generated 100 networks for each link utilization, e.g., 100 networks with the load of 10% on links, hence 1700 networks are generated for each network architecture shown in Fig. 8.4.

The next sub-sections present the results of the experiments. We compare the results of mapping the generated messages with LETRA and an intuitive mapping tool based on three different variables, including the link utilization, the number of messages, and the time it takes to schedule the ST messages.

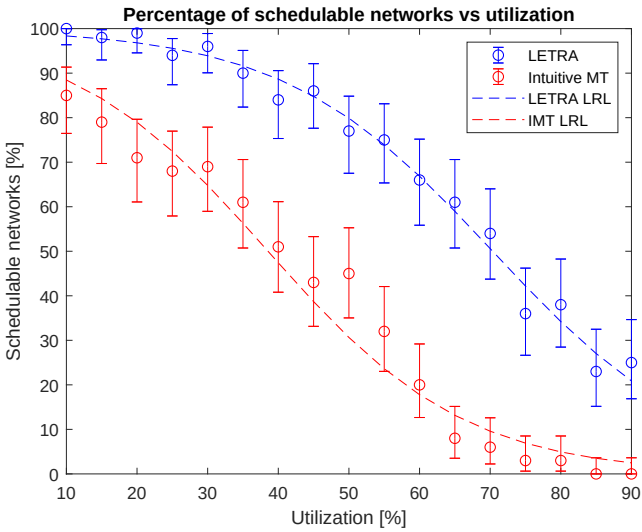


Figure 8.5: Percentage of schedulable networks with respect to the bandwidth utilization for the single-switch network where the schedulability percentage is represented by a circle with error bars and the dashed curve corresponds to its logistical regression tendency line (LRL).

Mean IMP [%]	Min IMP [%]	Max IMP [%]
86.65	37.52	149.47

Table 8.2: Performance improvements with respect to the percentage of bandwidth utilization for the single-switch network.

8.6.2 Results of the single-switch network

Fig. 8.5 shows the percentage of schedulable networks with respect to the utilization. The horizontal axis shows the utilization of the generated networks, while the vertical axis shows the percentage of networks that are schedulable with two different traffic mapping tools. The circles in the figure are the schedulable percentage of generated networks with specific network utilization and the error bars are calculated through the binomial analysis with 95% certainty. In addition, the dashed lines show the trend of the data as logistic regression. As it can be seen, LETRA results in more schedulable networks in all generated utilization compared to the intuitive mapping tool. For instance, when we generated traffic with 90% utilization on all links, which is a very high network utilization, LETRA gives just below 30% of the networks schedulable, whereas the intuitive mapping results in very few schedulable

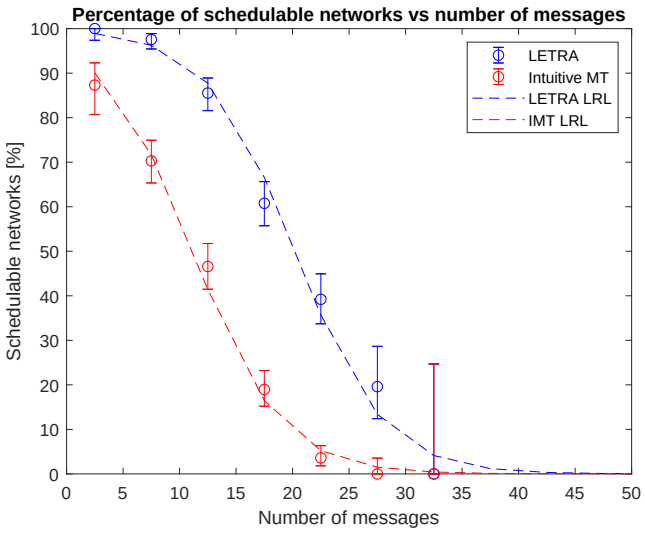


Figure 8.6: Percentage of schedulable networks with respect to the number of messages for the single-switch network where the schedulability percentage is represented by a circle with error bars and the dashed curve corresponds to its logistical regression tendency line (LRL).

networks. Table 8.2 shows the mean improvement by using LETRA compared to the intuitive mapping tool while Min IMP and Max IMP are the maximum and minimum possible improvements due to the result errors. LETRA results in 86.65% more schedulable networks compared to the intuitive mapping on average for the single-switch network architecture.

Fig. 8.6 shows the percentage of schedulable networks by varying the number of messages. Similar to the previous results, the vertical axis is the percentage of schedulable networks, and the horizontal axis is the number of generated messages. The figure also shows the error bars calculated through the binomial analysis with 95 % certainty and the tendency line calculated through logistic regression. Again, LETRA shows a significant improvement in the schedulability of networks compared to the intuitive mapping tool. For example, the intuitive mapping tool cannot schedule the networks when the number of messages is more than 35, however, LETRA can schedule a few of the generated networks up to 45 messages in the single-switch network. Table 8.3 shows the mean improvement between the two mapping tools, where it shows that on average 77.75% improvement by using LETRA.

Fig. 8.7 illustrates the time that it takes to schedule the ST messages in a generated message after mapping. In the figure, times are shown in ms and

Mean IMP [%]	Min IMP [%]	Max IMP [%]
77.57	35.93	120.85

Table 8.3: Performance improvements with respect to number of messages for the single-switch network.

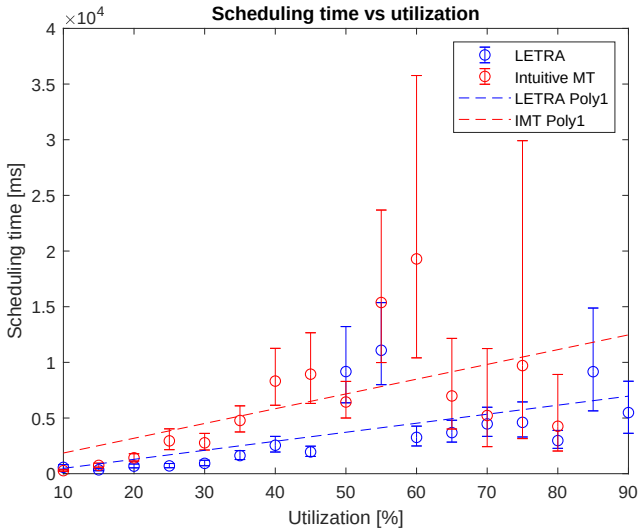


Figure 8.7: Scheduling time with respect to bandwidth utilization for the single-switch network where the scheduling time is represented by a circle with error bars and the dashed line corresponds to a linear regression (Poly1).

the error bars are calculated through the gamma distribution analysis with 95% certainty. We also used a linear trend line to show the overall trend of data. In this figure, the values present high variability and they do not follow any basic tendency curve due to the number of parameters that can affect the scheduling time, such as memory or CPU utilization, which could not be monitored during the execution of the experiments due to hardware limitations. However, it can show, in general, apart from having better performance, LETRA is also delivering the ST schedules faster compared to the intuitive mapping tool. The reason is that the intuitive mapping tool selects more messages to be ST messages, while LETRA decides based on many timing requirements which in general leads to less number of ST messages. This means that the legacy messages are not unnecessarily mapped into the ST class.

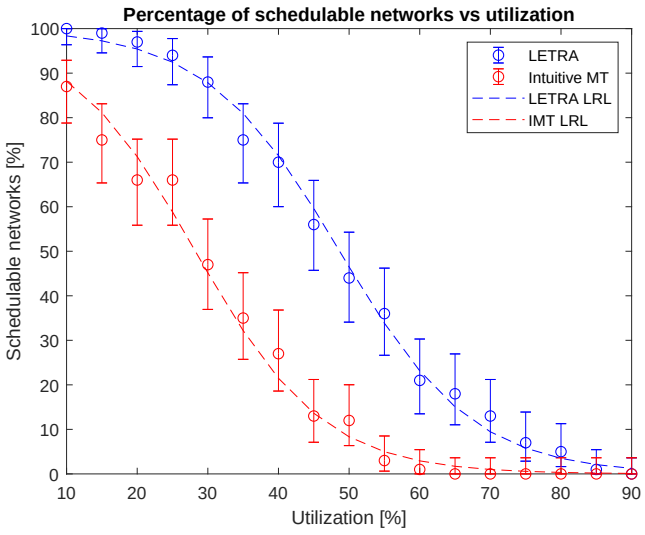


Figure 8.8: Percentage of schedulable networks with respect to the bandwidth utilization for the three-switch network where the schedulability percentage is represented by a circle with error bars and the dashed curve corresponds to its logistical regression tendency line (LRL).

Mean IMP [%]	Min IMP [%]	Max IMP [%]
90.74	32.30	165.28

Table 8.4: Performance improvements with respect to the percentage of bandwidth utilization for the three-switch network.

8.6.3 Results of the three-switch network

Fig. 8.8 shows the percentage of schedulable networks with respect to the bandwidth utilization for the three-switch network. Again, LETRA exhibits better performance compared to the intuitive mapping tool for all ranges of network utilization in larger networks. Table 8.4 shows mean improvement, which is 90.74% better performance with LETRA on average compared to the intuitive mapping tool. Although the performance of both mapping tools decreases with the size of the network, according to the results, we can conclude with 95% certainty that the percentage of improvement remains constant with the size of the network.

Fig. 8.9 shows the percentage of schedulable networks with respect to the number of messages. LETRA exhibits better performance compared to the intuitive mapping tool for the entire range of the number of messages ana-

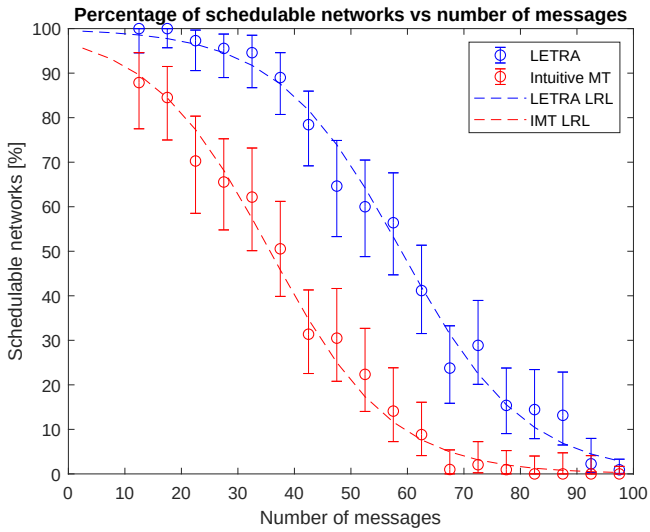


Figure 8.9: Percentage of schedulable networks with respect to the number of messages for the three-switch network where the schedulability percentage is represented by a circle with error bars and the dashed curve corresponds to its logistical regression tendency line (LRL).

Mean IMP [%]	Min IMP [%]	Max IMP [%]
83.41	27.19	157.84

Table 8.5: Performance improvements with respect to the number of messages for the three-switch network.

lyzed. Table 8.5 shows that the amount of improvement in the larger network is 83.41% on average when using LETRA to map the traffic.

Finally, Fig. 8.10 presents the time that it takes to schedule the ST messages after the mapping. Again, the results show that, besides LETRA having better performance, it is also faster in scheduling the ST messages compared to the intuitive mapping tool. Another interesting observation is that in the bigger network with high utilization, the ST messages can be scheduled within a reasonable time, while it is not the case if we map the traffic with the intuitive mapping tool where after 50% network utilization the ST messages cannot be scheduled. We believe that the reason is mainly due to the high amount of messages and utilization of the network.

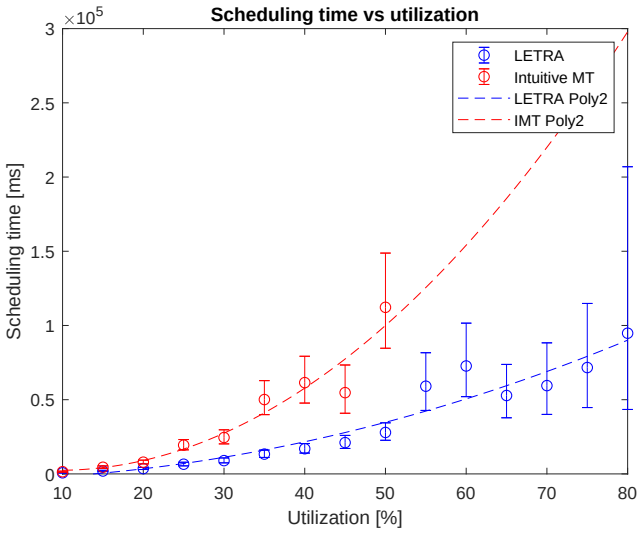


Figure 8.10: Scheduling time with respect to the bandwidth utilization for the three-switch network where the scheduling time is represented by a circle with error bars and the dashed curve corresponds to a quadratic regression (Poly2).

8.7 Conclusions and Future Work

We argued that one of the essential steps towards migrating from legacy Ethernet-based networks to TSN-based networks in industries is efficiently mapping the traffic into TSN traffic classes. Therefore, in this paper, we took a three-step strategy to achieve such a missing step. The steps include: (i) identifying the properties of legacy Ethernet-based messages by modeling them, (ii) map the Ethernet messages into different TSN classes, including ST, AVB, and BE classes, according to several timing properties, and (iii) develop a set of tools to evaluate the proposed mapping methodology, including the mapping tool, called LETRA, an ST scheduling tool and a schedulability analysis for the AVB messages. We also developed an intuitive mapping tool to show the performance of LETRA compared with that. We performed a set of experiments using two network architectures, being single-switch and three-switch architectures. We generated a set of messages randomly with a specific network utilization to show which mapping tool can result in more schedulable networks. The results show that in both network sizes LETRA performs much better, in concrete, 86.65% better performance in the smaller network and 90.74% in the larger network in average.

All these results were obtained for a specific network topology and traf-

fic configuration. In future work, we plan to run similar evaluations for other kinds of networks and schedulers to better define the mapping criteria and their performance. Moreover, we want to integrate it with the scheduler so the messages that can be placed in different traffic classes according to the mapping criteria can be specified while running the scheduler. On the other hand, we also plan to continue working on the other steps of the legacy integration. In this sense, we plan to develop a formal standard legacy Ethernet-based traffic model and adapt the schedulers to those message characteristics.

Bibliography

[1] IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams. *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pages C1–72, 2010.

[2] IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP). *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*, 2010.

[3] IEEE Standard for Local and Metropolitan Area Networks–Audio Video Bridging (AVB) Systems. *IEEE Std 802.1BA-2011*, pages 1–45, 2011.

[4] IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic. *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)*, pages 1–57, 2016.

[5] IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications. *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pages 1–421, 2020.

[6] 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.

[7] Inés Álvarez, Ignasi Furió, Julián Proenza, and Manuel Barranco. Design and experimental evaluation of the proactive transmission of replicated frames mechanism over time-sensitive networking. *Sensors*, 21(3):756, 2021.

[8] 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.

[9] Mohammad Ashjaei, Lucia Lo Bello, Masoud Daneshtalab, Gaetano Patti, Sergio Saponara, and Saad Mubeen. Time-sensitive networking in automotive embedded systems: State of the art and research opportunities. *Journal of Systems Architecture*, 110:1–47, September 2021.

[10] Mohammad Ashjaei, Gaetano Patti, Moris Behnam, Thomas Nolte, Giuliana Alderisi, and Lucia Lo Bello. Schedulability analysis of ethernet audio video bridging networks with scheduled traffic support. *Real-Time Systems*, 53(4):526–577, 2017.

[11] Voica Gavriluţ and Paul Pop. Traffic-type assignment for tsn-based mixed-criticality cyber-physical systems. *ACM Trans. Cyber-Phys. Syst.*, 4(2), 2020.

[12] Marina Gutiérrez, Wilfried Steiner, Radu Dobrin, and Sasikumar Punnekkat. Learning the parameters of periodic traffic based on network measurements. In *2015 IEEE International Workshop on Measurements & Networking (M&N)*, pages 1–6. IEEE, 2015.

[13] 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.

[14] Daniel Bujosa Mateu, Daniel Hallmans, Mohammad Ashjaei, Alessandro V Papadopoulos, Julian Proenza, and Thomas Nolte. Clock synchronization in integrated tsn-ethercat networks. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 214–221, 2020.

[15] Seifeddine Nsaibi, Ludwig Leurs, and Hans D Schotten. Formal and simulation-based timing analysis of industrial-ethernet sercos iii over tsn. In *2017 IEEE/ACM 21st International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–8, 2017.

[16] Niklas Reusch, Luxi Zhao, Silviu S Craciunas, and Paul Pop. Window-based schedule synthesis for industrial ieee 802.1 qbv tsn networks. In *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*, pages 1–4. IEEE, 2020.

[17] R. Salazar, T. Godfrey, L. Winkel, N. Finn, C. Powell, B. Rolfe, and M. Seewald. Utility Applications of Time Sensitive Networking White Paper (D3). Technical report, IEEE, 2018.

[18] S. Samii and H. Zinner. Level 5 by Layer 2: Time-Sensitive Networking for Autonomous Vehicles. *IEEE Communications Standards Magazine*, 2(2):62–68, 2018.

[19] Oscar Seijo, Zaloa Fernández, Iñaki Val, and Jesús A López-Fernández. SHARP: towards the integration of time-sensitive communications in

legacy LAN/WLAN. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–7, 2018.

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

[21] M. Wollschlaeger, T. Sauter, and J. Jasperneite. The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0. *IEEE Industrial Electronics Magazine*, 11(1):17–27, 2017.

Chapter 9

Paper B
HERMES: Heuristic Multi-queue Scheduler for TSN Time-Triggered Traffic with Zero Reception Jitter Capabilities.

Daniel Bujosa, Mohammad Ashjaei, Alessandro V. Papadopoulos, Julián Proenza, Thomas Nolte.
In Proceedings of the 30^{th} International Conference on Real-Time Networks and Systems (RTNS 2022).

Abstract

The Time-Sensitive Networking (TSN) standards provide a toolbox of features to be utilized in various application domains.The core TSN features include deterministic zero-jitter and low-latency data transmission and transmitting traffic with various levels of time-criticality on the same network. To achieve a deterministic transmission, the TSN standards define a time-aware shaper that coordinates transmission of Time-Triggered (TT) traffic. In this paper, we tackle the challenge of scheduling the TT traffic and we propose a heuristic algorithm, called HERMES. Unlike the existing scheduling solutions, HERMES results in a significantly faster algorithm run-time and a high number of schedulable networks. HERMES can be configured in two modes of zero or relaxed reception jitter while using multiple TT queues to improve the schedulability. We compare HERMES with a constraint programming (CP)-based solution and we show that HERMES performs better than the CP-based solution if multiple TT queues are used, both with respect to algorithm run-time and schedulability of the networks.

9.1 Introduction

Data communication in industrial systems has dealt with many challenges during recent years, such as scalability in data transmission, high volume of data exchange, the coexistence of diverse applications with different time-criticality requirements, and guaranteeing deterministic transmission for hard real-time traffic. These challenges are mainly due to recent demands for increasing functionalities in industrial systems that impose further pressure on the data communication design of such systems. For instance, in several application domains, e.g., autonomous vehicles and smart automation, many sophisticated smart sensors and cameras are utilized to perform newly added functionalities that require a high amount of communication bandwidth and at the same time meet their timing requirements. Besides the timing requirements, the rise of adaptive industrial systems imposes another criterion for designing data communication systems in which the network should be reconfigured due to changes in the environment. Therefore, in such systems, the configuration of the network is not seen as a one-time configuration in the initialization phase, but as a dynamic reconfiguration during the run-time (and operational) phase.

IEEE Audio-Video Bridging (AVB) Task Group (TG) was established in 2005 to provide Ethernet with soft real-time capabilities oriented to audio/video streaming. The three main projects developed by this TG are: (i) the IEEE Std 802.1AS [3] for clock synchronization, (ii) the IEEE Std 802.1Qav, which standardized the Credit-Based Shaper (CBS) [1]; and finally (iii) the IEEE Std 802.1Qat, which standardized the Stream Reservation Protocol (SRP) [2]. The latter standard is particularly interesting in the context of dynamic networks as it allows adding and removing streams at run-time. As the features that were developed by the AVB TG became relevant to other application areas, such as automotive [23], automation [27], and energy distribution [22], new requirements emerged. Therefore, in 2012, the TG broadened its objectives to meet the demands and was renamed to Time-Sensitive Networking (TSN) TG. Specifically, TSN TG's work was developed as a set of standards to provide transmission of hard and soft real-time traffic on the same network, deterministic zero-jitter and low-latency transmission, precise clock synchronization, fault tolerance mechanisms, and advanced network management allowing dynamic reconfiguration.

Motivation: One of the main features developed within TSN TG is the zero-jitter traffic transmission, known as the Time-Aware Shaper (TAS), which is particularly utilized in applications that require low-latency and low-jitter data transmission, e.g., in embedded control systems. The TAS allows transmission of Time-Triggered (TT) traffic while preventing any interference from

other traffic via a gate mechanism on the ports of the switches. Therefore, TAS requires the synthesis of the Gate Control Lists (GCL) that are specifying at which point in time each frame should be transmitted. A GCL is defined for each switch port which contains 8 queues, in such a way that the GCL identifies the moments in which the gate of each queue will be open. The scheduling of TT traffic, and its synthesis in GCLs, is known to be an NP-complete problem [19]. Several solutions are proposed in the literature to schedule TT traffic in TSN networks that are mainly based on Integer Linear Programming (ILP) and Constrained Programming (CP) [7]. These solutions are known to have high time complexity, i.e., they require a long time to schedule large networks, thus they are not generally scalable. In addition, these solutions are not suitable for systems that require dynamic reconfigurations as the new configuration should be created relatively fast. Few heuristic schedulers are also proposed, e.g., [17], whose performance is not properly compared with the ILP and CP solutions.

Paper contributions: In this paper, we propose a heuristic scheduler for TT traffic in TSN networks, called Heuristic Multi-queue Scheduler (HERMES), that takes advantage of multiple queues for TT traffic to provide high schedulability with very low scheduling times. Frames in HERMES can be configured to be scheduled in two modes of zero or relaxed reception jitter, which provides better control for users. Through a set of experiments, we show that HERMES can perform better than CP-based solutions, i.e., it results in more schedulable networks, by allowing it to use multiple queues, and at the same time, it provides the results within 17 to 800 times faster. In our experiments with two sizes of networks, we obtained schedules in less than $1ms$, which shows that HERMES is suitable for dynamic reconfiguration of networks.

Paper outline: The paper is organized as follows. Section 9.2 presents the related work. Section 9.3 presents the background. Section 9.4 presents the proposed algorithm, i.e., HERMES. Sections 9.5 analyzes the HERMES performance. Finally, Section 9.6 concludes the paper and indicates future directions.

9.2 Related Work

There have been many works on various TSN topics, including investigation of time-aware shaper mechanisms [4], proposing fault tolerance techniques [12], techniques to tolerate temporary faults in TSN networks with the use of retransmissions [5], and schedulability analysis of traffic with different TSN features [29], [14]. A recent comprehensive survey [7] presents the status of re-

search within TSN, including schedulability and scheduling problems, safety and security issues, and evaluation models and tools.

Within the context of TT traffic scheduling in TSN networks, the work in [26] present a scheduling algorithm formalized as an ILP while the works in [24] and [16] present a joint routing and scheduling algorithm formalized as an ILP and as a meta-heuristic scheduling approach based on a Genetic Algorithm (GA) approach, respectively. The work in [9] presents an SMT-based scheduler capable of scheduling networks with several TT queues. The work in [10] proposes a GCL synthesis approach based on Greedy Randomized Adaptive Search Procedure (GRASP) meta-heuristic [20], which takes AVB traffic into account, whereas the work in [11] proposes a joint routing and scheduling approach for TT and AVB traffic by means of an integrated heuristic and meta-heuristic strategy. In the latter work, the K-Shortest Path (KSP) method [28] is utilized for routing, and GRASP is used to schedule both TT and AVB at the same time. Moreover, the work in [8] synthesizes a network topology that supports seamless redundant transmission for TT traffic by proposing a greedy heuristic algorithm for joint topology, routing, and scheduling synthesis.

Protocol	**Routing**	**Multi-queuing**	**Schedule**	**ZRJ**
HLS	Yes	No	per frame	No
MML	Yes	No	per frame	No
BN	Yes	No	per frame	No
CV	Yes	No	per frame	No
MDP	Yes	No	per frame	No
HERMES	No	Yes	per link	Yes

Table 9.1: Comparison between heuristic schedulers.

The above-mentioned solutions are mostly based on ILP or constraint programming, while some of them exploit the use of meta-heuristics, e.g., GA. However, these solutions normally are highly time-complex, which makes them not scalable. Few works target heuristic solutions with lower time complexity. For instance, the work in [17] proposes a heuristic routing and scheduling algorithm called Heuristic List Scheduler (HLS) that is limited to a single TT queue, while the work in [25] compares 4 heuristic algorithms combining routing and scheduling (Modified Most Loaded Heuristic (MML), Bottleneck Heuristic (BN), Coefficient of Variation Heuristic (CV) [6][13], and Modified Dot Product Heuristic (MDP) [18]), all with scheduling times greater than 100 ms and unable to handle multiple queues. In this work, we propose a heuristic

algorithm, called HERMES, with scheduling times lower than 10 ms that uses multiple TT queues to improve schedulability. Moreover, the proposed algorithm provides two modes, one with zero reception jitter and relaxed reception jitter in the receiver end-station, see Section 9.3 for detailed description of zero and relaxed reception jitter. The zero reception jitter mode is configurable which is helpful for the applications in which the feature is not essential. Table 9.1 shows the main differences between the heuristic schedulers mentioned above, including HERMES. The features that are analyzed in this comparison include routing, multi-queuing for TT traffic, scheduling process, and support for zero reception jitter (ZRJ).

9.3 Background

TSN end-stations communicate by transmitting Ethernet frames through routes consisting of links and time-sensitive switches. In TSN, Ethernet frames belong to one of the eight possible priorities. The traffic is classified as one of the three available traffic classes, including TT traffic, AVB traffic, and BE traffic, where TT traffic has higher priority than other traffic classes and BE has the lowest priority. Note that several priorities may cover one traffic class, e.g., AVB can consist of classes A, B, and C, each associated with one priority level. A port of a TSN switch supports up to eight FIFO queues each of them associated with one priority level. Figure 9.1 shows an example of a time-sensitive device output port with four queues configured as TT traffic with the highest priority, AVB classes A and B traffic with the medium priority, and a BE traffic class as the lowest priority.

9.3.1 Time-Triggered Traffic

TT traffic is scheduled offline, which allows to know exactly in which time slot each TT frame is transmitted. This requires that interference between frames must be prevented. This is achieved through the TAS mechanism (see Figure 9.1). According to this mechanism, each queue has an associated gate that can be open or closed. The frames in a queue can be transmitted when the gate is open, otherwise, the frames are blocked for transmission.The gates are controlled by the GCL, which specifies at which point in time gates should be open, and it is a cyclic list that repeats the schedule. The time that gates are open or closed can be specified at the nanosecond level for each entry of the GCL and we refer to the opening time of a gate as *window*.

Figure 9.2 shows an example of TAS operation for two TT queues with two different priorities, i.e., priority 6 and 7. In this example, we assume that

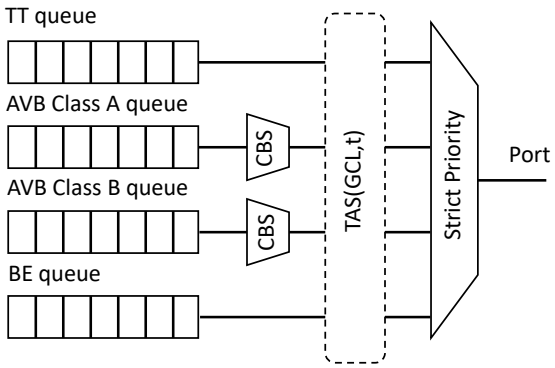


Figure 9.1: A TSN egress port with four FIFO queues: one TT queue, two AVB queues, and one BE queue.

three TT frames with a period of 4 time units are transmitted through a switch port where one of the TT frames is set to the highest priority 7 (blue frame), while the other two frames (red and green) are set to priority 6. As the periods of the frames are equal, the hyper-period (the least common multiple) of them is 4 time units. Therefore, the GCL cycle is defined as 4 time units allowing gates operation in each time slot and repeating every 4 time units. According to the schedule in this example, which is set in the GCL, at time T0 till T1 the gate for priority 6 queue is open (shown as 1 in GCL), whereas the gate for priority 7 is closed (shown as 0 in GCL) allowing transmission of the red frame. Further, in the time slot between T1 and T2, the blue frame can be transmitted as the gate for priority 7 is open. Between T2 and T3 both gates are closed, thus no transmission can occur, and finally, the gate of priority 6 queue is open in the last time slot that allows the transmission of the last frame, i.e., the green frame. Two cycles of frame transmissions are represented at the bottom of Figure 9.2.

On the other hand, TSN supports multi-hop communication which imposes other restrictions when scheduling. For a frame to be transmitted on a link, it must have been previously transmitted through the preceding links in the route of the frame. Furthermore, the order of frames in transmission is also important. Considering that the queues in the switches are FIFO, if a frame arrives to a switch before another one, it will also be transmitted first. Figure 9.3 shows an example of a multi-hop schedule. The figure shows three switches (S1, S2, and S3) and three links, two of them connecting S1 and S2 with S3 and one in the S3 output. Two frames are exchanged between these 3 switches, one blue frame is sent by S1 and one red frame is sent by S2 and both frames are

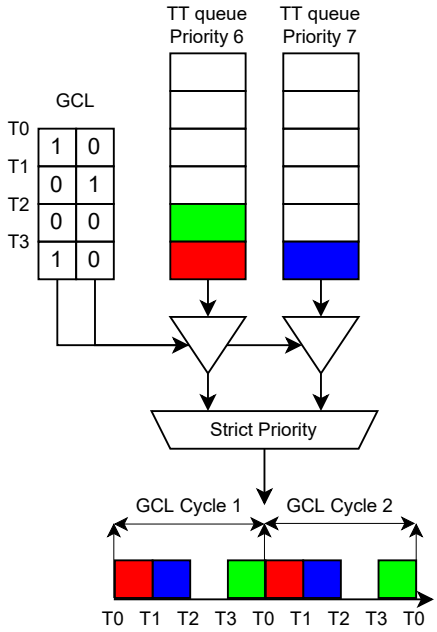


Figure 9.2: TSN TAS gate mechanism.

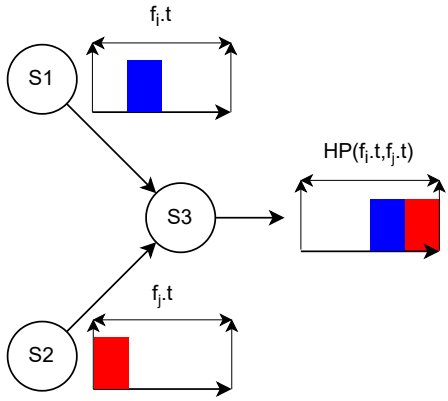
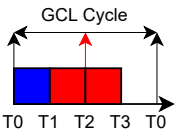
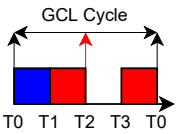


Figure 9.3: Multi-hop behavior of TSN and GCLs.

forwarded by S3 through the output link. As we can see, both are sent after they are received by S3. However, according to the schedule that is decided for this case (for whatever the reason), the red frame arrives before the blue frame to S3 but the blue frame is sent by S3 before the red frame, thus this schedule would not be possible with a single queue. In this case, the red and blue



(a) Schedule of two frames with and without reception jitter.



(b) Schedule of two frames, both with zero reception jitter.

Figure 9.4: Difference between reception jitter and zero reception jitter.

frames must have been assigned to different queues and hence have different priorities.

Finally, interference-free transmission of frames ensures their zero jitter transmission. This means that the variations in the transmission and reception of each frame with respect to the schedule will be zero, assuming that the clock drift is zero. However, in this work, not only the jitter but also the reception jitter has been considered. The reception jitter is defined as the variability of the instant of reception by the receiver end-station of a frame with respect to its period. For example, Figure 9.4 shows the schedule over the GCL cycle of a TSN output port connected to the input port of the receiving end-station. In this schedule two frames are shown, a blue frame with a period of 4 time units and a red frame with a period of 2 time units. According to the schedule example in Figure 9.4a, the blue frame has zero reception jitter since it is always transmitted at time unit 0, whereas the red frame has reception jitter since it is transmitted at time unit 1 in its first instance (T1) of the hyper-period and at time 0 in its second instance (T2). Zero reception jitter is particularly interesting in heterogeneous systems combining TSN components and legacy components that cannot adopt TSN synchronization mechanisms. In these cases, a frame with zero reception jitter helps the synchronization of the applications even if the devices are not properly synchronized. Throughout the paper, we denote zero reception jitter by ZRJ and reception jitter by RJ. Note that in this case, as shown in Figure 9.4b, it would be enough to delay the transmission of the second instance of the red frame by one time unit for both frames to have ZRJ.

9.3.2 AVB and BE Traffic

AVB frames are not scheduled offline, i.e., they are scheduled via CBS once they arrive at the switch port. The gates are normally open for them unless TT traffic is to be transmitted. The CBS aims at improving the Quality-of-Service (QoS) of lower priority traffic while ensuring a minimum of bandwidth

utilization. Finally, BE frames have no real-time guarantees, thus they will be transmitted when their gate is open and the CBS is negative for all higher priority traffic.

9.4 Proposed scheduling algorithm

We developed what we call Heuristic multi-queue Scheduler (HERMES) which generates the global schedule for the transmission of TT traffic. Our goal is to reduce the scheduling time while achieving high schedulability through the use of different numbers of TT queues and providing zero jitter. Moreover, HERMES can provide zero reception jitter.

HERMES calculates the GCL of each egress port of the end-stations and TSN switches. To do this, unlike other heuristic algorithms that schedule frame by frame, our algorithm decides the schedule link by link (and then for each link deciding the schedule of each frame to be transmitted in that link) starting with the destination links and ending with the source links scheduling each frame as late as possible according to their timing constraints. The reason why we design HERMES to calculate the schedule link by link instead of frame by frame is that in this way the conflicts between frames are detected before the entire frame has been scheduled. On the contrary, when the network is scheduled frame by frame, the schedule of one frame may hinder the scheduling of the other frames and the algorithm will have to schedule that frame again throughout all its links. On the other hand, links are scheduled from destination to source because the destination link is the most restrictive link especially if the frame has more restrictive reception time constraints such as frames that must be received with zero reception jitter. In addition, in each links, each frame is scheduled as late as possible so that the preceding links have enough time margin between the frame's period start and the offset of the same frame in the last scheduled link. However, this does not imply that the frames will have the highest possible latencies because an improvement as simple as looking for the minimum offset of all frames and moving all frames earlier by that amount can be applied.

The use of multiple queues for TT traffic to improve the schedulability is thoroughly explained in Section 9.4.2. However, it is worth to mention that HERMES does not consider relative priorities between TT queues, i.e., all queues have the same priority. The isolation of frames by only opening the gate of a single queue at a time eliminates the arbitration among TT frames which is performed by the Strict Priority module (see Figure 9.1) if more than one queue has its gate open. The queues are only used to help the scheduler meeting the order condition explained in Section 9.3.

Although HERMES uses only unicast frames, the algorithm would work equally well with multicast frames. As we will see in the algorithm description, HERMES only schedules those links whose frames have already been scheduled in the subsequent links. In this way, in the case of multicast frames, when scheduling the link before the fork, the following links will already be scheduled and therefore the algorithm can continue to function normally only taking into account the offset and restrictions of several following links instead of a single following one.

9.4.1 System Model

In this work the communication network model consists of two main sets, one for the links $\mathcal{L}$ and one for the TT-frames $\mathcal{F}_{\text{TT}}$. Each link $l \in \mathcal{L}$ is unidirectional, is defined by its identifier, and has a parameter $l.\phi$ indicating in which phase of the execution of HERMES the link will be scheduled. Indeed, the execution of the scheduling algorithm presented in this paper is divided into phases, with a total of Φ phases. In the case of full-duplex links, two links with opposite directions are instantiated. On the other hand, each frame $f \in \mathcal{F}_{\text{TT}}$ is characterized by seven parameters $f = \langle t, w, d, q, u, n, \mathcal{S} \rangle$, i.e.,

1. the period $f.t$,
2. the length of the frame or the size of the window needed to transmit the frame $f.w$,
3. the deadline $f.d$,
4. the queue of the frame in all egress ports of the whole route $f.q$,
5. a parameter to decide in which order the frames in the links are scheduled, which in this case is the frame utilization $f.u$ (see Algorithm 2 in Section 9.4.2),
6. the number of links in the route $f.n$, and
7. a set $f.\mathcal{S}$ containing the route and the schedule of each link in the route.

Each element $s \in f.\mathcal{S}$ includes three parameters $s = \langle \zeta, \iota, \mathcal{O} \rangle$:

1. the link $s.\zeta$ of the route assigned in reverse order, i.e. $s_1.\zeta$ being the destination link and $s_{f.n}.\zeta$ the source link,
2. the number of instances $s.\iota$ of the frame in the link, and
3. a set $s.\mathcal{O}$ indicating the offset of each instance according to the period start of the instance.

9.4.2 HERMES

The input to the scheduler consists of a list of TT frames characterized by their period, frame length, deadline, and routing expressed as a vector of unidirectional link identifiers. With this list, HERMES executes the following steps to obtain the schedule.

Algorithm 1: HERMES - DivPhases

Data: $\mathcal{L}, \mathcal{F}_{\text{TT}}$
Result: $\forall l \in \mathcal{L}$ return $l.\phi$
procedure *DivPhases*
  for $\forall l \in \mathcal{L}$ **do** $l.\phi \leftarrow NULL$ **end**
  $\Phi \leftarrow 1$
  while $\exists l \in \mathcal{L} : l.\phi = NULL$ **do**
    for $f_i.s_j, f_k.s_x \in \mathcal{F}_{TT}, f.\mathcal{S} : f_i.s_{j-1}! = NULL \wedge f_k.s_{x-1} = NULL$ **do**
      if $\exists! f_i.s_j.\zeta = f_k.s_x.\zeta$ **then**
        $f_i.s_j.\zeta.\phi \leftarrow \Phi$
      end
    end
    $\Phi \leftarrow \Phi + 1$
  end

Algorithm 2: HERMES - AssignFrameUtilization

Data: $\mathcal{F}_{\text{TT}}$
Result: $\forall f \in \mathcal{F}_{\text{TT}}$ return $f.u$
procedure *AssignFrameUtilization*
  for $\forall f \in \mathcal{F}_{TT}$ **do**
    $f.u \leftarrow \frac{f.w}{f.d} \cdot f.n$
  end

Algorithm 1, DivPhases: First of all, as mentioned before, links are divided into phases where all frames in all links assigned to that phase are scheduled togther. The only condition for a link l_i to be assigned to a phase $l_i.\phi$ is that all frames transmitted through that link $f \in \mathcal{F}_{\text{TT}} : f.s_j.\zeta = l_i$ must have all previous links (links closer to destination) assigned to previous phases $\forall f.s_k.\zeta : k < j | f.s_k.\zeta.\phi < f.s_j.\zeta.\phi$. To do that, if there are links not assigned to any phase, the algorithm adds a new phase and checks if links not

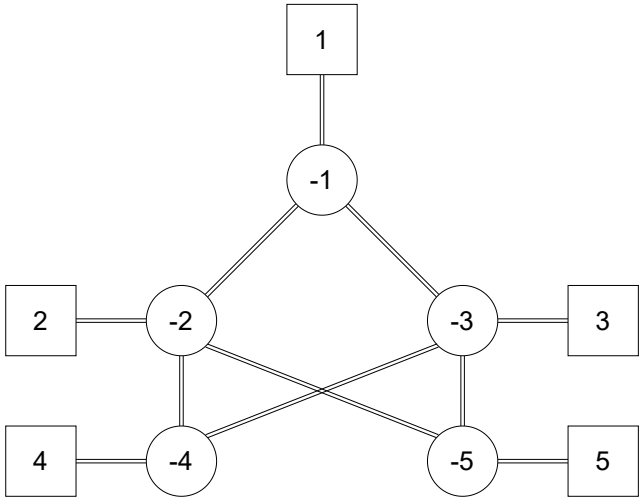


Figure 9.5: TSN Network example.

assigned are ready to be assigned in the new phase according to previous conditions. Note that, as mentioned before, links in the route are ordered from destination to source, i.e., routes are scheduled backwards and in each phase all links which, fulfilling the above conditions, are independent can be scheduled. Figure 9.5 presents a network example where the squares with positive numbers represent end-stations and the circles with negative numbers represent switches. For this example, Table 9.2 shows a list of frames and their corresponding routes expressed as a list of pairs of end-station/switch identifiers indicating the link and its direction. Moreover, Table 9.3 shows how links in reverse order (from destination to source) are assigned to phases and how these links are delayed (represented by arrows) in the scheduling process until the preceding links are assigned to a previous phase. Finally, for this example, Table 9.4 shows the resulting distribution of the links in phases. The reason for scheduling the links in reverse order is that, due to the need for determinism in the reception link, this link becomes the most restrictive. This is particularly important to obtain a zero reception jitter schedule as mentioned in Section 9.4. On the other hand, note that, although this scheduler can be used in feed-forward networks, the routes cannot present triple dependencies in a loop. For example, the routes of the frames in Table 9.5 cannot be distributed in phases. The reason is that, as it can be seen in Table 9.6, the triple dependency in the loop causes a deadlock in the DivPhases algorithm as some links in the route will be indefinitely delayed.

Algorithm 3: HERMES – Schedule

Data: $\mathcal{L}, \mathcal{F}_{\text{TT}}$
Result: $\forall f \in \mathcal{F}_{\text{TT}}$ return $f.S$

procedure *Schedule*
  for $\forall f \in \mathcal{F}_{\text{TT}}$ **do** $f.q \leftarrow 1$ **end**
  for $p = 1..\Phi$ **do**
    for $\forall l \in \mathcal{L} : l.\phi = p$ **do**
      $HP \leftarrow \text{LCM}(\{f.t : f.s.\zeta = l\})$
      for $\forall f.s \in f.\mathcal{S} : f.s.\zeta = l$ **do** $f.s.\iota \leftarrow \frac{HP}{f.t}$ **end**
      for $\forall f \in \mathcal{F}_{TT}, f.s_x \in f.\mathcal{S} : f.s_x.\zeta = l$ *from highest to lowest* $f.u$ **do**
        for $i = f.s_x.\iota..1$ **do**
          $f.s_x.o_i \leftarrow \min(f.s_{x-1}.\mathcal{O}, f.d) - f.w$
          while $Collision(f.s_x.o_i) \vee Order1(f.s_x.o_i) \vee Order2(f.s_x.o_i)$ **do**
            if $Collision(f.s_x.o_i)$ **then**
              $f.s_x.o_i \leftarrow f.s_x.o_i - 1$
            else
              if $Order1(f.s_x.o_i)$ **then**
                $f.q \leftarrow f.q + 1$
                if $f.q > Q$ **then**
                  $f.s_x.o_i \leftarrow f.s_x.o_i - 1$
                end
              end
              if $Order2(f.s_x.o_i)$ **then**
                $f.q \leftarrow f.q + 1$
                if $f.q > Q$ **then**
                  return Unschedulable
                end
              end
            end
            if $f.s_x.o_i < 0$ **then**
              return Unschedulable
            end
          end
        end
      end
    end
  end

Frame	Route
f1	1 -1 ; -1 -2 ; -2 2
f2	2 -2 ; -2 -1 ; -1 -3 ; -3 3
f3	2 -2 ; -2 -4 ; -4 4
f4	3 -3 ; -3 -5 ; -5 5
f5	3 -3 ; -3 -1 ; -1 1
f6	3 -3 ; -3 -4 ; -4 -2 ; -2 2
f7	4 -4 ; -4 -2 ; -2 -1 ; -1 1
f8	5 -5 ; -5 -2 ; -2 -1 ; -1 1
f9	5 -5 ; -5 -3 ; -3 3

Table 9.2: Example of frame routes for Figure 9.5

Frame	ϕ 1	ϕ 2	ϕ 3	ϕ 4	ϕ 5	ϕ 6
f1	-2 2	-1 -2	1 -1			
f2	-3 3	-1 -3	-2 -1	2 -2		
f3	-4 4	-2 -4	$\rightarrow$	2 -2		
f4	-5 5	-3 -5	$\rightarrow$	$\rightarrow$	$\rightarrow$	3 -3
f5	-1 1	-3 -1	$\rightarrow$	$\rightarrow$	$\rightarrow$	3 -3
f6	-2 2	$\rightarrow$	$\rightarrow$	-4 -2	-3 -4	3 -3
f7	-1 1	$\rightarrow$	-2 -1	-4 -2	4 -4	
f8	-1 1	$\rightarrow$	-2 -1	-5 -2	5 -5	
f9	-3 3	-5 -3	$\rightarrow$	$\rightarrow$	5 -5	

Table 9.3: DivPhases procedure

Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6
-1 1	-1 -2	1 -1	2 -2	-3 -4	3 -3
-2 2	-1 -3	-2 -1	-4 -2	4 -4	
-3 3	-2 -4		-5 -2	5 -5	
-4 4	-3 -1				
-5 5	-3 -5				
	-5 -3				

Table 9.4: Resulting link distribution in phases

Algorithm 2, AssignFrameUtilization: To decide which frames in the link will be scheduled first, the utilization of the frame along its route is calculated according to the formula $f.u = \frac{f.w}{f.d} \cdot f.n$. This formula can change

Frame	Route
f1	1 -1 ; -1 -3 ; -3 -4 ; -4 4
f2	3 -3 ; -3 -4 ; -4 -2 ; -2 -1 ; -1 1
f3	4 -4 ; -4 -2 ; -2 -1 ; -1 -3 ; -3 3

Table 9.5: Example of frame routes with feed-forward dependencies for Figure 9.5

Frame	ϕ 1	ϕ 2	ϕ 3	ϕ 4
f1	-4 4	$\rightarrow$	$\rightarrow$	...
f2	-1 1	$\rightarrow$	$\rightarrow$	...
f3	-3 3	$\rightarrow$	$\rightarrow$	...

Table 9.6: Infinite DivPhases procedure

depending on the needs. However, in this case, we sought to prioritize the frames that required the most bandwidth at specific time periods (between the period start and deadline) as this causes the free space for scheduling to be consumed very quickly, so it is important to prioritize them to provide them with the space they need.

Algorithm 3, Schedule (l.2-8): Firstly, we initialize the queues by assigning queue 1 to all frames. Then, HERMES goes through all the phases and, within each phase, all the links, and calculates the hyper-period (HP) by calculating the Least Common Multiple (LCM) of all frames of each link and the number of instances of each frame in the link by dividing the HP by the frame period $f.t$. In this way, HERMES schedules phase by phase, where in each phase the links can be scheduled in parallel because they are independent. Finally, in each link, the schedule of the frames is done in order of descending $f.u$, instance by instance from last ($f.s_x.\iota$) to the first instance in the HP. This is to prevent, in case of having frames with deadlines bigger than periods, later instances to be scheduled before previous instances.

Algorithm 3, Schedule (l.9-10): Secondly, we initialize the offset of each instance of the frame in the link to the minimum between the deadline of the frame and the release of the frame in the previous link (the following link if we consider order from source to destination) if any. Then we check if the offset assigned to the instance of the frame makes it collide with another previously scheduled instance and if the order of reception and transmission in the switch between this link and the previous one is adequate. **Algorithm 3, Schedule**

(l.11-34): Finally, if there is a collision, defined as

$$\begin{aligned} &Collision(f_i.s_j.o_k) = \exists f_m.s_j.o_n : \\ &[f_m.s_j.o_n + f_m.t \cdot (n-1) \leq f_i.s_j.o_k + f_i.w + f_i.t \cdot (k-1)] \wedge \\ &[f_m.s_j.o_n + f_m.w + f_m.t \cdot (n-1) \geq f_i.s_j.o_k + f_i.t \cdot (k-1)], \end{aligned} \tag{9.1}$$

the frame moves backward until it encounters an empty space. Once an empty space is found, the reception and transmission order in the switch is checked. If the frame arrives at the switch later than another frame with which shares the transmission link and which is also transmitted later than the frame under scheduling in the shared transmission link, i.e.,

$$\begin{aligned} &Order1(f_i.s_j.o_k) = \exists f_m.s_j.o_n : \\ &[f_m.s_{j-1}.\zeta = f_i.s_{j-1}.\zeta] \wedge \\ &[f_m.s_j.o_n + f_m.t \cdot (n-1) < f_i.s_j.o_k + f_i.t \cdot (k-1)] \wedge \\ &\left[f_m.s_{j-1}.o_{n'} + f_m.t \cdot (n'-1) > f_i.s_{j-1}.o_{k'} + f_i.t \cdot (k'-1)\right] \end{aligned} \tag{9.2}$$

the switch must change the queue or move backward in the schedule so that it arrives earlier than the frame instance in order conflict. On the other hand, if the frame arrives at the switch earlier than another frame that shares the transmission link and is transmitted earlier than the frame under scheduling in the shared transmission link, i.e.,

$$\begin{aligned} &Order2(f_i.s_j.o_k) = \exists f_m.s_j.o_n : \\ &[f_m.s_{j-1}.\zeta = f_i.s_{j-1}.\zeta] \wedge \\ &[f_m.s_j.o_n + f_m.t \cdot (n-1) > f_i.s_j.o_k + f_i.t \cdot (k-1)] \wedge \\ &\left[f_m.s_{j-1}.o_{n'} + f_m.t \cdot (n'-1) < f_i.s_{j-1}.o_{k'} + f_i.t \cdot (k'-1)\right] \end{aligned} \tag{9.3}$$

the only option is to change the queue or the network configuration will be unschedulable for the ordering approach used. Moreover, if the offset becomes negative due to the frame advance, the configuration will also be unschedulable. Note that changing the queue is not enough, it is also necessary to check that such a change does not affect the order conditions of the previously scheduled links but for the sake of simplicity, this has not been included in the algorithm.

On the other hand, in HERMES frames can be configured as RJ or ZRJ. For the sake of simplicity, this is not reflected in the algorithm but it consists of forcing all offsets to be equal on the reception link of the frames configured as ZRJ. If a frame is configured as RJ, it can be received by the receiver at different points in time even if reception is deterministic. For example, if one

frame is scheduled as RJ and has a period of 4s, HERMES may schedule it in a loop of 3 instances where the first instance has an offset of 1s, the second instance has an offset of 3s, and the third instance has an offset of 2s. In this way, the instances of this frame will be received at seconds 1, 7, 10, 13, 19, and so on. However, if a frame is configured as ZRJ, it will be received by the receiver at a constant rate equal to the period. For example, if one frame is scheduled as ZRJ and has a period of 4s, HERMES schedules it in a way that the offsets of all instances are the same. In this way, if the frame has an offset of 2s the frame will be received at seconds 2, 6, 10, 14, 18, and so on. As mentioned before, this behavior is especially interesting for legacy devices that cannot implement TSN synchronization protocols but want to execute applications in a TT fashion, or want to exchange their own legacy synchronization frame with other legacy devices through TSN.

9.5 Evaluation of HERMES

9.5.1 Experimental setup

For the evaluation of HERMES, in this paper, we used the LETRA evaluation toolset (ETS) developed in [15]. LETRA ETS provides a set of integrated tools capable of performing automated experiments regarding the scheduling and schedulability analysis of TSN networks. In this section, we will explain LETRA ETS and the modifications that have been done for this paper. The reader is referred to [15] for more information about LETRA ETS.

The toolchain of the LETRA ETS used in this work is depicted in Figure 9.6. The main input to the ETS is the network configuration, including the network topology and traffic characteristics. For this paper, we use two network topologies both following a line-star topology. The network topologies are depicted in Figure 9.7 and they are a small single-switch network consisting of 3 nodes (S1) and a larger three-switch network consisting of 9 nodes (S3).

For the traffic characteristics, we set the traffic to be only TT as we exclude the effect of HERMES on lower priority traffic in this stage. The network bandwidth is set to 10Mbps to prevent generating too many frames when reaching loads around 90% utilization to avoid taking more than a week to conduct each round of experiments due to the CP scheduler and the network generator which are the two most time-consuming tools. The maximum number of generated frames is set to 100, however, depending on the selected utilization, the frame number can be different. The frame length is selected within the range [500,1000] Bytes. The minimum and maximum allowed periods are set

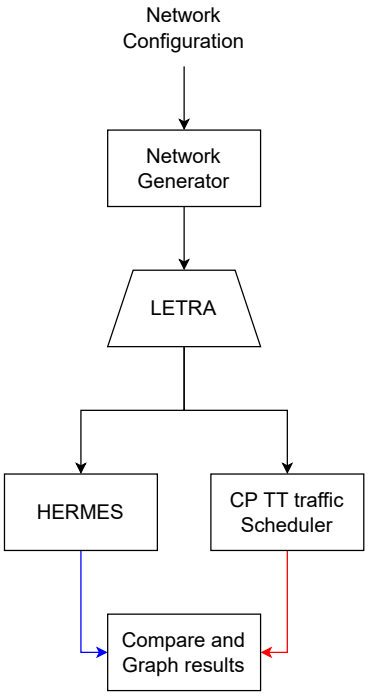


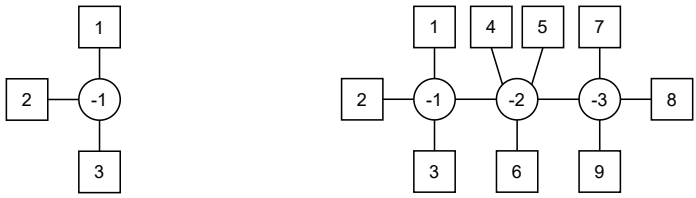
Figure 9.6: LETRA evaluation tool set modification.

to 200μs and 1000μs respectively, while deadlines were assigned the same values as the corresponding periods. Note that the frames will be generated such that the utilization of all links will be the one selected as an input, e.g., when we select 10% utilization the traffic generator selects the frame sizes and routes to obtain 10% on all links if possible.

The first step of LETRA ETS is generating random traffic. We used the network configuration as input of the network generator to generate 1700 sets of TT traffic randomly for each network topology (100 TT traffic configurations for each of the values of the utilization, which are taken [from 10% to 90% in steps of 5% of utilization]).

The next step is LETRA, as it can be seen in Figure 9.6, which is a mapping tool to map the generated traffic into TSN traffic classes, i.e., TT, AVB and BE. In this paper, we are only interested in TT traffic, thus, we skip the traffic mapping. However, ETS is integrated in a way that the output of each tool is the input of the next. For this reason, we used LETRA only as a translator between the output of the network generator and the input of the schedulers.

Finally, each generated network processed by LETRA is used as input to



(a) Single-switch network (S1). **(b)** Three-switches network (S3).

Figure 9.7: Experimental network topologies.

both HERMES and a CP scheduler [21]. The CP scheduler runs in the only available mode which is with one queue and RJ while HERMES, depending on the experiment, is configured with up to 4 TT queues and with the frames configured in either mode RJ or ZRJ.

Once the scheduling of all the generated networks is done, we compare the two schedulers, i.e., HERMES and CP, with respect to the time that it takes for each of them to give a solution and the number of networks for which each of the schedulers is able to find a schedule (number of networks considered as schedulable by each scheduler). The experiments are done for different values of the network utilization, which is the same on all network links, e.g., 10% utilization is considered in all links of the network. We show the results for different network utilization in several graphs in the following sections. In the graphs that show the number of schedulable networks, the circles indicate the percentage of networks generated that could be scheduled while error bars represent the error in the percentage obtained through the binomial analysis with 95% certainty. Additionally, these graphs include dashed trend lines obtained through logistic regression adjustment to present the trend of changes. Moreover, Table 9.7 compares the schedulability between HERMES in modes RJ and ZRJ and the CP scheduler on networks S1 and S3. In the table, we analyze schedulability $S(u)$, as a function of the utilization u, for the range of utilizations under analysis ($u \in [10\%, 90\%]$). Since the schedulability $S(u)$ is sampled, we approximate it by a logistic regression, that we indicate with the notation $\widehat{S}(u)$. Then we define the accumulated schedulability for a network x as:

$$AS_x = \int_{0.1}^{0.9} \widehat{S}_x(u)du \tag{9.4}$$

which we use to compare schedulers using the accumulated scheduling ratio defined as:

$$ASR_{x,y}[\%] = \frac{AS_x}{AS_y} \cdot 100 \tag{9.5}$$

to measure the percentage of schedulable networks of x compared to y. On

the other hand, in the graphs that show the time taken by each scheduler to give a solution, the circles show the average time needed to get the schedule in milliseconds for each utilization level, while error bars are calculated using the gamma distribution with 95% certainty. Moreover, the graphs include a dashed trend line obtained by fitting the data to an exponential function or a polynomial function of order 1 or 2.

9.5.2 Results of the scheduling time

We start with the evaluation by analyzing the time it takes to give a schedule for both network topologies. In Figs. 9.8 and 9.9 scheduling times for all HERMES modes and the number of queues as well as the scheduling time of the CP scheduler for networks S1 and S3 respectively are shown. In both graphs, we can see how the scheduling time of the CP scheduler is exponentially increasing with the percentage of utilization and the number of frames while HERMES remains with scheduling times below 10ms. This implies that HERMES exhibits scheduling times from tens of times lower than the CP scheduler to thousands of times lower for high utilization values. Furthermore, we can see how for 50% utilization the scheduling time in the S1 network for the CP scheduler is 3000ms while for the S3 network it is 8000ms, which also shows a large increase in scheduling time with the size of the network.

On the other hand, Figs. 9.10 and 9.11 show a detail of the scheduling times specifically for HERMES in RJ mode for networks S1 and S3 respectively. Both graphs show an increase in scheduling time proportional to the square of the utilization, which is related to the number of frames. On the other hand, the scheduling time is proportional to the longest path between two end-stations. In this case, as shown in Figure 9.7, the ratio between the longest routes is 4/2. For a utilization of 60%, we observe that in the S1 network HERMES with 2 and 3 queues takes 1 and 2 ms respectively while in the S3 network for the same number of queues the scheduling time is 2 and 4 ms, which corresponds to the ratio calculated above. Finally, it is also possible to identify that scheduling time doubles with every extra queue, for example, for a 60% utilization in the S3 network, HERMES takes 2, 4, and 8 ms to get a schedule with 2, 3, and 4 queues respectively. Although the complexity doubles with each extra queue used, the fact that the queues are limited to 8 reduces its impact and allows HERMES to remain scalable.

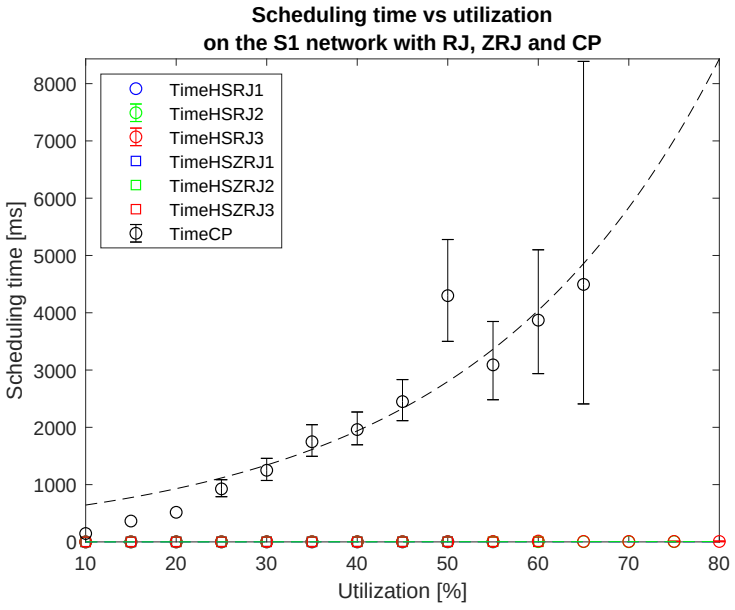


Figure 9.8: Scheduling time for different levels of network utilization on network S1 of a CP scheduler and HERMES with and without zero reception jitter with 1, 2 and 3 queues.

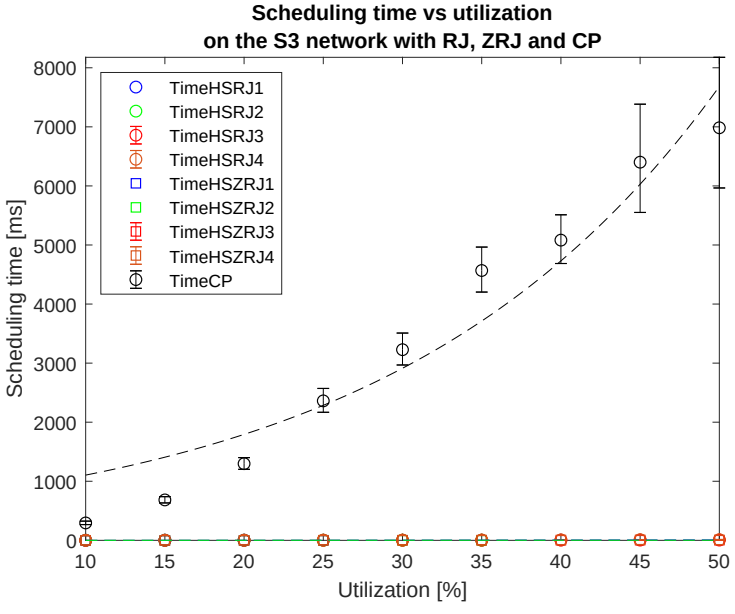


Figure 9.9: Scheduling time for different levels of network utilization on network S3 of a CP scheduler and HERMES with and without zero reception jitter with 1, 2, 3 and 4 queues.

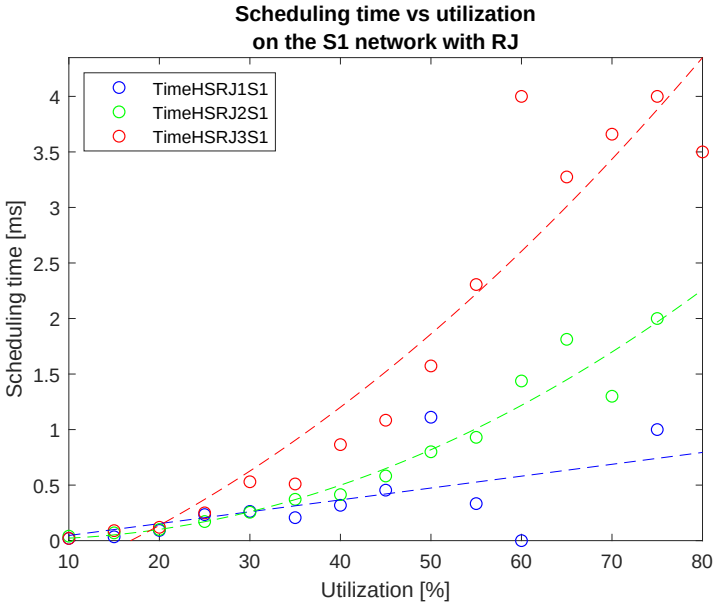


Figure 9.10: Scheduling time for different levels of network utilization on network S1 of HERMES with reception jitter with 1, 2 and 3 queues.

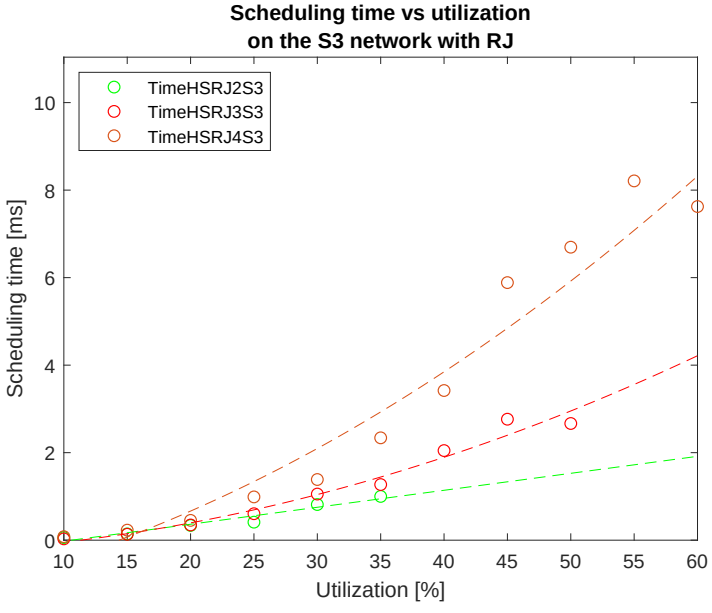


Figure 9.11: Scheduling time for different levels of network utilization on network S3 of HERMES with reception jitter with 2, 3 and 4 queues.

9.5.3 Results of the schedulability

Figure 9.12 shows in black the schedulability of the CP scheduler for a single-switch network (S1) with the set of generated networks. On the other hand, in blue, green, and red we can see the HERMES schedulability with 1, 2, and 3 TT queues respectively in mode RJ (with jitter in the reception). The first observation is that it would be enough to increase the number of queues to 2 to obtain the same schedulability as the CP scheduler with 1 queue but, in addition, with three queues it is even possible to exceed by more than 32% the schedulability achieved by the CP scheduler, as shown in the first column of Table 9.7. These results, together with those shown in the previous subsection, show the usefulness of HERMES in contexts where the number of queues is not a constraint but the schedulability time is, e.g., run-time configurations.

Figure 9.13 shows the HERMES schedulability in zero reception jitter (ZRJ) mode for the S1 network. This graph shows that in this case, from 2 queues onwards, the schedulability stagnates due to the tough constraint that the ZRJ mode imposes. However, in the second column of Table 9.7 it can be seen how the schedulability is lower than in RJ mode, except for the case of one queue where the ZRJ mode restriction facilitates the scheduling of certain cases. Since, in general, the schedulability in ZRJ mode is lower than the schedulability in RJ mode, it is recommended to limit this mode to frames that really require it.

Figure 9.14 shows in black the schedulability of the CP scheduler for a three-switches network (S3) with the set of generated networks. On the other hand, in blue, green, red, and orange we can see the HERMES schedulability with 1, 2, 3, and 4 TT queues respectively in mode RJ. In this graph, we can see how HERMES scales worse the longer the route of the frame, being necessary up to 4 queues to surpass the schedulability levels that are attainable with the CP scheduler, as shown in the third column of Table 9.7. However, it can also be noticed that with 3 queues 81% schedulability is achieved so the im-

	Network S1		Network S3	
N° Q	$ASR_{RJ,CP}$	$ASR_{ZRJ,RJ}$	$ASR_{ZRJ,CP}$	$ASR_{ZRJ,RJ}$
1	51.04	101.58	17.37	102.59
2	98.50	80.54	55.09	104.62
3	131.99	62.73	81.59	96.33
4	–	–	101.35	81.98

Table 9.7: Schedulability comparison between HERMES with different number of queues (Q) in mode RJ and ZRJ and the CP scheduler on networks S1 and S3.

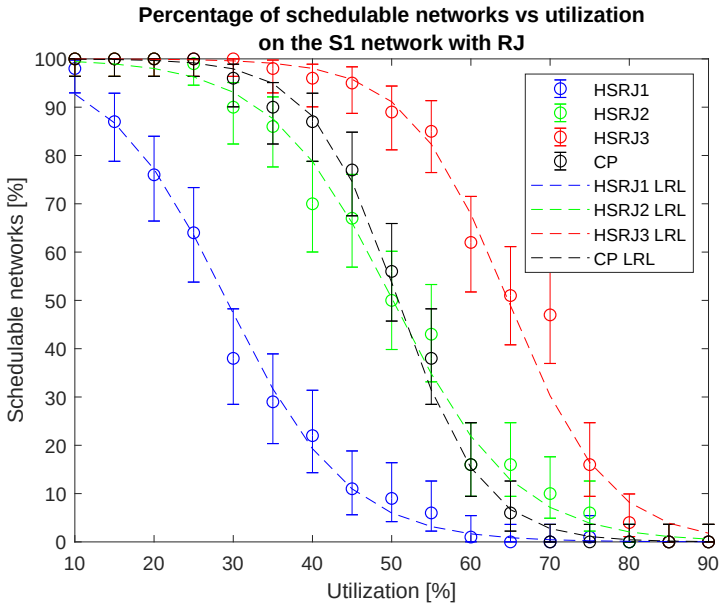


Figure 9.12: Schedulability for different levels of network utilization on network S1 of a CP scheduler and HERMES allowing reception jitter with 1, 2 and 3 queues.

provement in scheduling time can still be worthwhile. For example, different approaches can be tried to order the frames, apart from frame utilization, so that, although each has a lower schedulability, together can cover all the cases covered by the CP scheduler even in less time since different frame scheduling orders in the links will provide different schedules.

Figure 9.15 shows the HERMES schedulability in ZRJ mode for the S3 network. Similar to what happened in the S1 network, in this case, the schedulability in ZRJ mode also stagnates. However, the stall occurs after the third queue. On the other hand, as we can see in the fourth column of Table 9.7, with fewer queues the ZRJ constrain may slightly improve schedulability but for more queues, the difference is greater and increasing so, since more queues are needed to achieve high values of schedulability for this kind of traffic, again, this mode should be left for very specific frames if high schedulability wants to be achieved.

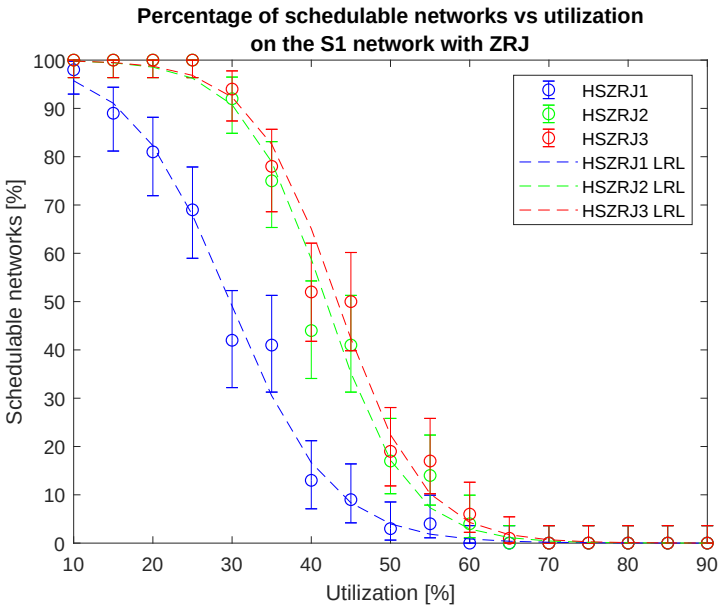


Figure 9.13: Schedulability for different levels of network utilization on network S1 of HERMES in zero reception jitter mode with 1, 2 and 3 queues.

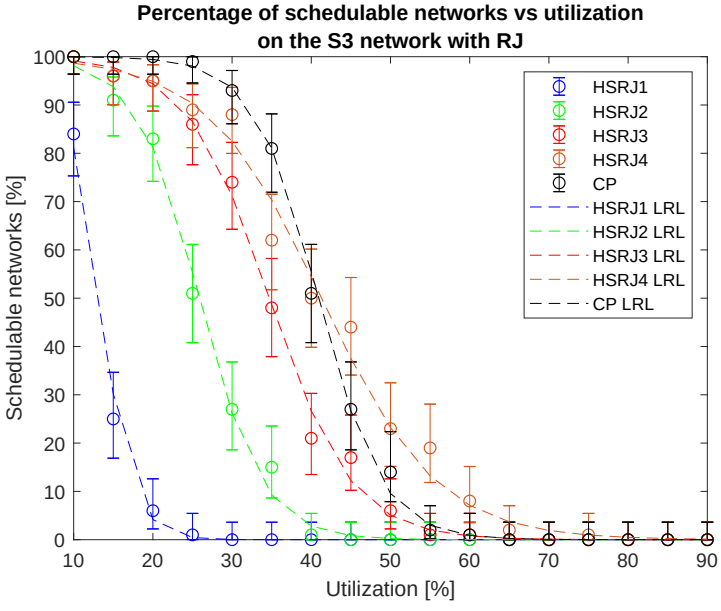


Figure 9.14: Schedulability for different levels of network utilization on network S3 of a CP scheduler and HERMES allowing reception jitter with 1, 2, 3 and 4 queues.

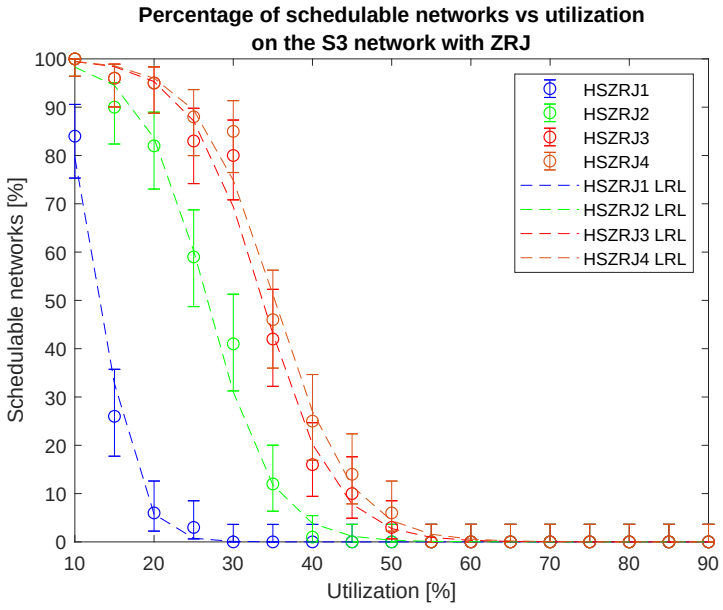


Figure 9.15: Schedulability for different levels of network utilization on network S3 of HERMES in zero reception jitter mode with 1, 2, 3 and 4 queues.

9.6 Conclusion and Future Work

We argued that developing fast scheduling algorithms are crucial specially for adaptive and evolutionary systems. Therefore, in this work, we have developed a fast heuristic scheduler for TT traffic in TSN networks called HERMES that can match the level of schedulability of reference schedulers by using several TT queues. We also use the LETRA ETS to evaluate HERMES performance showing that by using several queues HERMES can outperform the schedulability of CP schedulers with a single queue but with HERMES exhibiting scheduling times of less than 10 ms, which implies that HERMES is hundreds or thousands of times faster. In addition, HERMES supports the integrative capability of TSN by providing a more restrictive ZRJ mode that facilitates the integration into TSN networks of legacy devices that cannot implement TSN's own synchronization mechanisms.

In this work, we focus on TT traffic scheduling. However, in previous works, we have developed a TSN mapping tool and an AVB analyzer. Therefore, the next step is to integrate all these tools to create a toolset capable of mapping and scheduling traffic taking into account the real-time requirements of all kinds of traffic.

Bibliography

[1] IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams. *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pages C1–72, 2010.

[2] IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP). *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*, 2010.

[3] IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications. *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pages 1–421, 2020.

[4] 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.

[5] Inés Álvarez, Ignasi Furió, Julián Proenza, and Manuel Barranco. Design and experimental evaluation of the proactive transmission of replicated frames mechanism over time-sensitive networking. *Sensors*, 21(3):756, 2021.

[6] Emmanuel Arzuaga and David R Kaeli. Quantifying load imbalance on virtualized enterprise servers. In *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, pages 235–242, 2010.

[7] Mohammad Ashjaei, Lucia Lo Bello, Masoud Daneshtalab, Gaetano Patti, Sergio Saponara, and Saad Mubeen. Time-sensitive networking in automotive embedded systems: State of the art and research opportunities. *Journal of Systems Architecture*, 110:1–47, September 2021.

[8] Ayman A Atallah, Ghaith Bany Hamad, and Otmane Ait Mohamed. Fault-resilient topology planning and traffic configuration for ieee 802.1 qbv tsn networks. In *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, pages 151–156. IEEE, 2018.

[9] Silviu S Craciunas, Ramon Serna Oliver, Martin Chmelík, and Wilfried Steiner. Scheduling real-time communication in ieee 802.1 qbv time sen-

sitive networks. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, pages 183–192, 2016.

[10] Voica Gavriluţ and Paul Pop. Scheduling in time sensitive networks (tsn) for mixed-criticality industrial applications. In *14th IEEE International Workshop on Factory Communication Systems (WFCS)*, pages 1–4. IEEE, 2018.

[11] Voica Gavriluţ, Luxi Zhao, Michael L Raagaard, and Paul Pop. Avb-aware routing and scheduling of time-triggered traffic for tsn. *IEEE Access*, 6:75229–75243, 2018.

[12] 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.

[13] Leonard Kleinrock. Queueing systems, volume i: Theory, vol. i, 1975.

[14] Lucia Lo Bello, Mohammad Ashjaei, Gaetano Patti, and Moris Behnam. Schedulability analysis of time-sensitive networks with scheduled traffic and preemption support. *Journal of Parallel and Distributed Computing,*, 144,, 2020.

[15] Daniel Bujosa Mateu, Mohammad Ashjaei, Alessandro V Papadopoulos, Julian Proenza, and Thomas Nolte. Letra: Mapping legacy ethernet-based traffic into tsn traffic classes. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2021.

[16] Maryam Pahlevan and Roman Obermaisser. Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks. In *2018 IEEE 23rd international conference on emerging technologies and factory automation (ETFA)*, volume 1, pages 337–344. IEEE, 2018.

[17] Maryam Pahlevan, Nadra Tabassam, and Roman Obermaisser. Heuristic list scheduler for time triggered traffic in time sensitive networks. *ACM Sigbed Review*, 16(1):15–20, 2019.

[18] Rina Panigrahy, Kunal Talwar, Lincoln Uyeda, and Udi Wieder. Heuristics for vector bin packing. *research. microsoft. com*, 2011.

[19] Michael Lander Raagaard and Paul Pop. Optimization algorithms for the scheduling of ieee 802.1 time-sensitive networking (tsn). *Tech. Univ. Denmark, Lyngby, Denmark, Tech. Rep*, 2017.

[20] Mauricio GC Resende and Celso C Ribeiro. Grasp: Greedy randomized adaptive search procedures. In *Search methodologies*, pages 287–312. Springer, 2014.

[21] Niklas Reusch, Silviu S Craciunas, and Paul Pop. Dependability-aware routing and scheduling for time-sensitive networking. *arXiv preprint arXiv:2109.05883*, 2021.

[22] R. Salazar, T. Godfrey, L. Winkel, N. Finn, C. Powell, B. Rolfe, and M. Seewald. Utility Applications of Time Sensitive Networking White Paper (D3). Technical report, IEEE, 2018.

[23] S. Samii and H. Zinner. Level 5 by Layer 2: Time-Sensitive Networking for Autonomous Vehicles. *IEEE Communications Standards Magazine*, 2(2):62–68, 2018.

[24] Eike Schweissguth, Dirk Timmermann, Helge Parzyjegla, Peter Danielis, and Gero Mühl. Ilp-based routing and scheduling of multicast realtime traffic in time-sensitive networks. In *2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–11. IEEE, 2020.

[25] Ammad Ali Syed, Serkan Ayaz, Tim Leinmüller, and Madhu Chandra. Dynamic scheduling and routing for tsn based in-vehicle networks. In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2021.

[26] Marek Vlk, Kateřina Brejchová, Zdeněk Hanzálek, and Siyu Tang. Large-scale periodic scheduling in time-sensitive networks. *Computers & Operations Research*, 137:105512, 2022.

[27] M. Wollschlaeger, T. Sauter, and J. Jasperneite. The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0. *IEEE Industrial Electronics Magazine*, 11(1):17–27, 2017.

[28] Jin Y Yen. Finding the k shortest loopless paths in a network. *management Science*, 17(11):712–716, 1971.

[29] Luxi Zhao, Paul Pop, Zhong Zheng, and Qiao Li. Timing analysis of avb traffic in tsn networks using network calculus. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 25–36, 2018.

Chapter 10

Paper C
Clock Synchronization in Integrated TSN-EtherCAT Networks.

Daniel Bujosa, Daniel Hallmans, Mohammad Ashjaei, Alessandro V. Papadopoulos, Julián Proenza, Thomas Nolte.
In the 25^{th} IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2020).

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.

10.1 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] [9]. 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.

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

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

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.
- 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 10.2 describes the basics of clock synchronization in both TSN and EtherCAT. Section 10.3 presents the related work. Section 10.4 formulates the problem, while Section 10.5 proposes our solution. Then, utilizing a formal modeling framework, Section 10.6 evaluates the solution. Finally, Section 10.7 concludes the paper and gives future directions.

10.2 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.

10.2.1 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.

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

10.2.1.1 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.

10.2.1.2 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.

10.2.1.3 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. 10.1.

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

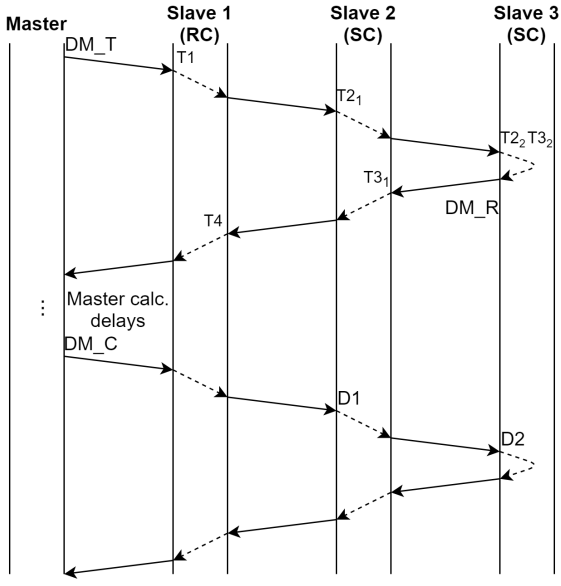


Figure 10.1: Delay Measurement Mechanism.

reference clock and the slave clocks is calculated by the EtherCAT master node by: $\frac{T4-T1-T3_i+T2_i-LDRC-LDSC_i}{2}$.

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.

10.2.2 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 (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.

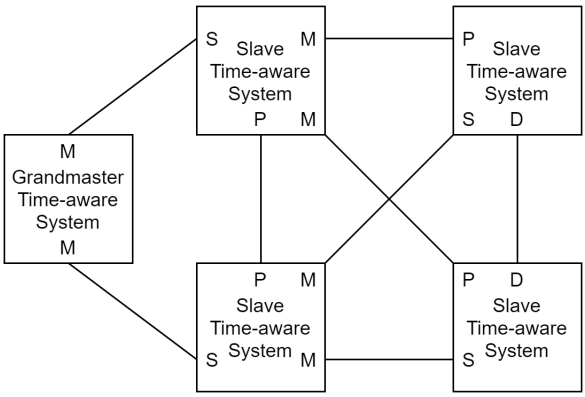


Figure 10.2: Example of TSN time-synchronization spanning tree.

10.2.2.1 BMCA

The BMCA constructs a time-synchronization spanning tree with the grandmaster as the root. One example of this can be seen in Fig. 10.2. 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. 10.3, 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.

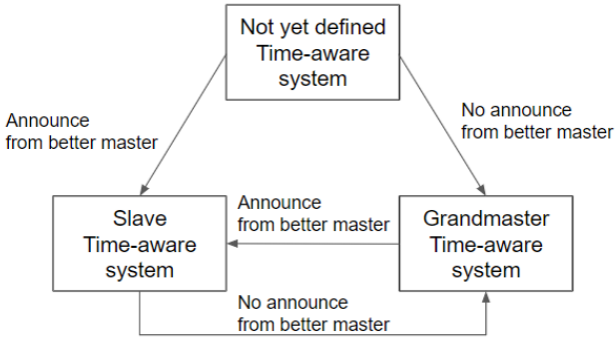


Figure 10.3: Time-aware system BMCA evolution.

10.2.2.2 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. 10.4. 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 $Delay = \frac{(T4-T1)-(T3-T2)}{2}$.

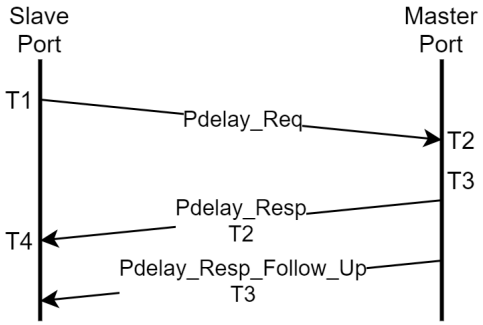


Figure 10.4: PDM diagram.

10.2.2.3 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.

10.3 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 [18], scheduling policies [10] and load balancing in TSN networks [3]. 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

[4] `https://1.ieee802.org/tsn/`
[5] `https://www.ethercat.org`

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 [19] 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 [13, 20, 5]. Several works addressed the problem of improving clock synchronization in EtherCAT devices from different point of view. In this context, the work presented in [15] actively measures the synchronization error and compensates the error with a proposed mechanism. Moreover, the work presented in [16] proposed to use a central oscillator to coordinate the clocks among the nodes. The work in [12] 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 [11, 7, 8, 17]. By improving the master node, e.g., by replacing it with a special-purpose hardware, the precision of $20ns$ can be achieved [14].

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.

10.4 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

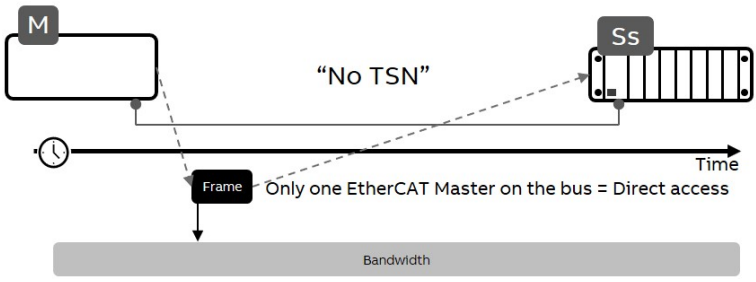
and EtherCAT networks, two key challenges must be considered, including stream adaption and clock synchronization. The former challenge is necessary to be addressed, e.g. by adding EK1000 hardware[6], or by providing adaption directly in the TSN bridge port connected to the EtherCAT devices, such that the TSN network will be able to handle the EtherCAT telegrams, as pointed out in Section 10.3. 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. 10.5 shows three scenarios of connecting a master node (M) to multiple slave nodes (Ss) in an EtherCAT network. In Fig. 10.5a, the master node is directly connected to the slave nodes, while in Figs. 10.5b and 10.5c the EtherCAT nodes are connected through a TSN network. The difference between the two latter scenarios is if clock synchronization is present (Fig. 10.5c) or not (Fig. 10.5b). 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. 10.5b.

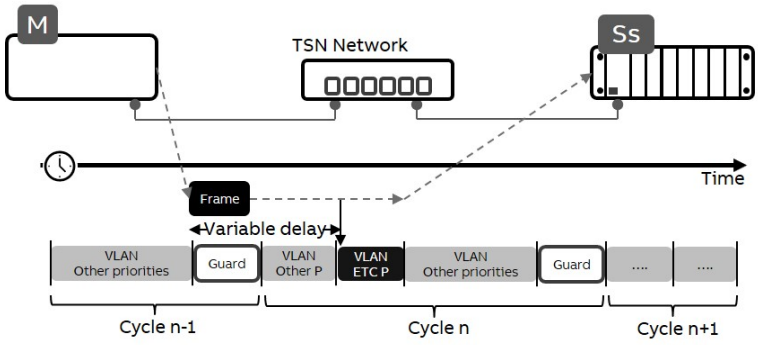
The jitters that are added to a closed control loop can, in the worst case, result in a control system that is unstable and thereby unable to provide the designed efficiency. More details in this regard will be described in Section 10.5. However, even if the introduced jitter caused by lack of clock 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.

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. 10.5c. The jitter that is caused by inte-

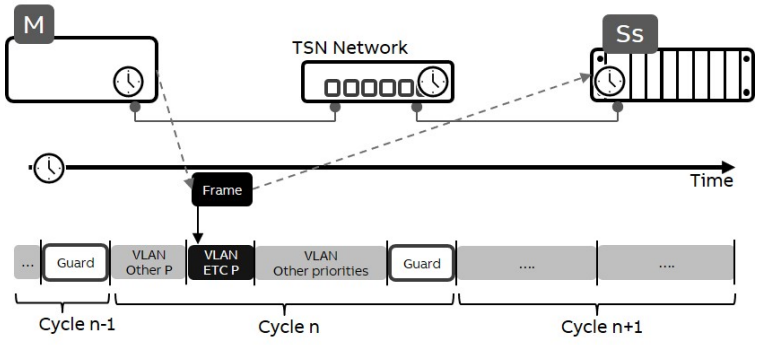
[6] `https://www.beckhoff.com`



(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.

Figure 10.5: EtherCAT network behaviors depending on the presence of a TSN network, possibly with clock synchronization.

grating 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, with-

out 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 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.

10.5 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.

10.5.1 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 10.2) 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,

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

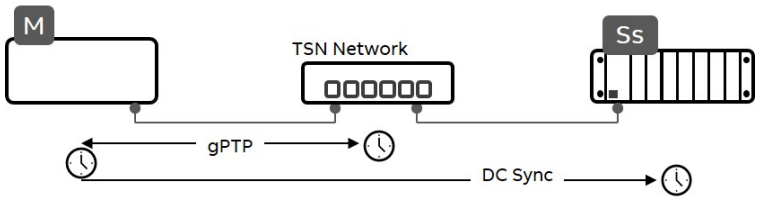


Figure 10.6: Clock synchronization protocols and hierarchy in an integrated TSN-EtherCAT network.

as both TSN and EtherCAT have similar precision, in the order of hundred nanoseconds. This precision will be maintained throughout the whole system. Fig. 10.6 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 10.2.2, 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.

10.5.2 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 [1]. 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 10.6.3.

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 10.2.1, 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 10.6). 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.

10.6 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.

10.6.1 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 10.5, 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) [4], 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

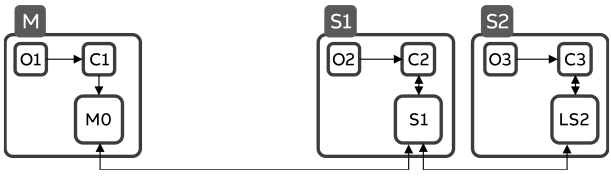
of different automata using *channels*. Channels are special variables that can force two or more automata to take a specific edge at the same time. UPPAAL provides a formal query language that can be used to specify the properties that we want to check in the model. These queries have two parts: state 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.

10.6.2 UPPAAL models

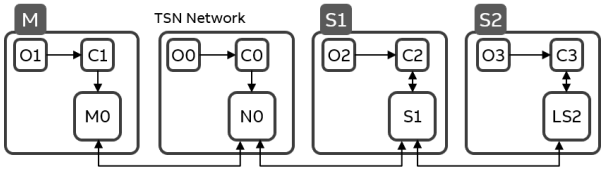
We have created 3 different UPPAAL models [8]. All of them modeled an EtherCAT network in which the reference clock is located in the EtherCAT master node, as described in the proposed solution in Section 10.5. In addition, all three EtherCAT networks consist of one EtherCAT master and two slave nodes. In the following, we present a brief overview of the models and queries that are developed for evaluation purposes.

The first model M1 (Fig. 10.7a) 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 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.

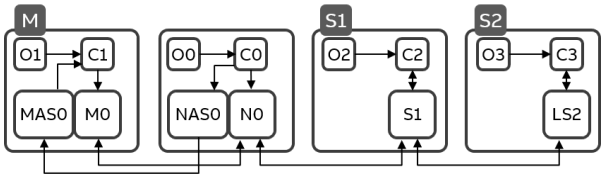
[8]The UPPAAL models are publicly available at `https://github.com/DanielBujosa/ETFA2020_TSN_EtherCAT_ClockSync.git`.



(a) UPPAAL M1 model.



(b) UPPAAL M2 model.



(c) UPPAAL M3 model.

Figure 10.7: TSN-EtherCAT clock synchronization integration UPPAAL models.

The second model M2 (Fig. 10.7b) 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. 10.7c) 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 10.5. 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.

10.6.3 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 10.2.1 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 10.5, 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 10.2.1, 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.

10.6.4 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 10.2.1 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 10.1: 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 10.1 and Table 10.2, 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 10.2: 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 10.1, 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 10.2, 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 10.1 is 25%, whereas in Table 10.2 it is 60%. However, these high values are due to the abstractions applied to the model, which are described in Section 10.6.2. 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.

10.7 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.

Bibliography

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

[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] 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.

[4] Gerd Behrmann, Alexandre David, and Kim G. Larsen. *A Tutorial on* UPPAAL, pages 200–236. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[5] G. Cena, I. C. Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino. Evaluation of EtherCAT distributed clock performance. *IEEE Trans. Industrial Informatics*, 8(1):20–29, 2012.

[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*, pages 3398–3403, 2010.

[7] M. Cereia, I. C. Bertolotti, and S. Scanzio. Performance of a real-time EtherCAT master under Linux. *IEEE Trans. Industrial Informatics*, 7(4):679–687, 2011.

[8] 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)*, pages 744–748, 2017.

[9] D. Jansen and H. Buttner. Real-time ethernet the EtherCAT solution. *Computing Control Engineering Journal*, 15(1):16–21, 2004.

[10] 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.

[11] 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)*, pages 527–533, 2018.

[12] 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)*, pages 1369–1374, 2017.

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

[14] 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)*, pages 1–8, 2013.

[15] S. Park, H. Kim, H. Kim, C. N. Cho, and J. Choi. Synchronization improvement of distributed clocks in EtherCAT networks. *IEEE Communications Letters*, 21(6):1277–1280, 2017.

[16] 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)*, pages 1–8, 2019.

[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*, pages 2418–2422, 2010.

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

[19] Xuepei Wu and Lihua Xie. End-to-end delay evaluation of industrial automation systems based on EtherCAT. In *IEEE Conf. Local Computer Networks (LCN)*, pages 70–77, 2017.

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

Chapter 11

Paper D
Improved Clock Synchronization in TSN Networks with Legacy End-Stations.

Daniel Bujosa, Mohammad Ashjaei, Alessandro V. Papadopoulos, Julián Proenza, Thomas Nolte.
Technical report at Mälardalen University, Sweden, pending for submission to a journal.

Abstract

In order to facilitate the adoption of Time Sensitive Networking (TSN) by the industry, it is necessary to develop tools to integrate legacy systems with TSN. In this paper, we propose a solution for the coexistence of different time domains from different legacy systems with their corresponding synchronization protocols in a single TSN network. To this end, we experimentally identified the effects of replacing the communications subsystem of a legacy Ethernet-based network with TSN in terms of synchronization. Based on the results, we propose a solution called TALESS (TSN with Legacy End-Stations Synchronization). TALESS is able to identify the drift between the TSN communications subsystem and the integrated legacy devices (end-stations) and modify the TSN schedule to adapt to the different time domains to avoid the effects of the lack of synchronization between them. We validate TALESS through both simulations and experiments on a prototype. Thereby we demonstrate that thanks to TALESS, legacy systems are able to synchronize through TSN and even improve features such as their reception jitter or their integrability with other legacy systems.

11.1 Introduction

Since the creation of the IEEE Time Sensitive Networking (TSN) Task Group (TG) in 2012, industry interest in TSN has not stopped growing. TSN seems to be essential for the incipient Industry 4.0 [18] as well as of interest in various areas such as automotive [13] and energy distribution [12]. The reason behind this growing interest is that TSN establishes a set of standards to provide deterministic zero-jitter and low-latency transmission, fault tolerance mechanisms, advanced network management allowing dynamic reconfiguration, precise clock synchronization, and flexibility in traffic transmission. The latter property is particularly relevant to the adoption of TSN in the industry. The flexibility in the traffic transmission allows the transmission of different types of traffic over the same physical links, which enables the migration of all kinds of legacy traffic to TSN. This in turn facilitates the adoption of TSN by the industry as many of the legacy devices and implemented solutions could be kept, which would reduce adoption time and costs.

Most current networks are composed of different sub-networks each with different communication protocols to meet their specific requirements. This hinders communication between sub-networks and therefore their integrability, as well as it increases the complexity of the overall network due to the use of different technologies, cabling redundancy, etc. Thanks to the flexibility of TSN traffic, it is possible to combine different types of traffic in the same network, which facilitates the communication and integration of the sub-networks. This integration can be done in different ways, such as through the use of gateways. However, this would not allow sub-networks to take advantage of other TSN features such as higher bandwidth or low jitter. Therefore, we propose to directly replace the communications subsystem of the legacy network, i.e. the set of devices exclusively responsible for communication excluding the end-stations, with TSN but in such a specific way that the legacy end-stations can maintain their behavior and communication protocols (including their legacy synchronization protocol) agnostic to the change. This improves the integration of the different legacy systems as well as it allows them to benefit from the enhancements of TSN. However, certain types of TSN traffic, such as Time-Triggered (TT) traffic, require the path from the source to the destination, including all switches in the network, to be synchronized since TT traffic is transmitted according to a fixed time schedule. This requirement is not fulfilled in many existing industrial networks where non-TSN nodes (*end-stations* in TSN terminology) do not feature TSN synchronization mechanisms and due to their hardware or software limitations may not even be able to support them. For the remainder of the paper, we will use the term *legacy network*

for the original network, i.e. before replacing the communications subsystem with TSN. On the other hand, the term *legacy end-stations* will be used in reference to the nodes of legacy networks that have been integrated by replacing their communications subsystem with TSN; while the term *legacy system* will refer to the set of end-stations that were originally part of the same legacy network and are synchronized through the legacy synchronization protocol of the legacy network, sharing a common time view.

The key piece to achieving the above-indicated integration of the legacy end-stations with the new TSN communication subsystem is a novel mechanism we propose in this paper. This mechanism is called TALESS (TSN with Legacy End-Stations Synchronization) and it is devised to prevent the negative effects resulting from the lack of synchronization between the TSN communications subsystem and the legacy systems integrated with it. TALESS transparently improves network performance without requiring modifications to legacy systems. Preventing any modifications in the legacy end-stations makes TALESS a general solution that allows applying the proposed integration approach on any TSN network where several Ethernet-based legacy systems, with different communication protocols, communicate. We verify it, on the one hand, using a model that simulates long executions (1 year) of a communication network. This model simulates the behavior of the TSN network with and without TALESS for different types of legacy end-station transmissions. Finally, we implement TALESS in a network prototype by which we experimentally verify both the solution and its simulation model. However, in our experiments, we exaggerated certain clock parameters of the legacy system to magnify the effects of the solution and thereby be able to demonstrate its behavior in a reasonable run-time.

Contributions. Among different requirements in integrating legacy systems onto a TSN network, an essential component is clock synchronization to maintain proper behavior of the communication among devices, especially if these devices require TT traffic transmissions. Thus, the main target of this paper is the development of a mechanism to avoid the adverse effects of carelessly putting together legacy systems with TSN in terms of clock synchronization. The main contributions of this paper are as follows:

- We identify problems caused by the lack of synchronization through experiments on a network prototype.
- We propose a mechanism, named TALESS, to remove the effects of lack of synchronization when including legacy systems into a TSN network.
- We model TALESS to validate the effectiveness of the proposed solution in a simulation environment with realistic network values.

- Finally, we implement TALESS in a network prototype to experimentally showcase its impact on utilizing legacy systems in a TSN network. We also compare the results of the experiment with the simulation model to validate both the solution and the simulation.

Outline. The paper is organized as follows. Section 11.2 presents the related work. Section 11.3 provides the necessary background for the understanding of this paper. Section 11.4 presents the effects of including legacy systems into a TSN network in terms of clock synchronization. Section 11.5 proposes TALESS. Section 11.6 presents the simulation model and experimental setup used to validate TALESS, while Section 11.7 presents the results obtained from both the simulations of the model and the experiments on the prototype. Finally, Section 11.8 concludes the paper and presents future directions.

11.2 Related Work

One of the most crucial aspects of TSN technology is clock synchronization. However, to the best of our knowledge, there is no work that provides a solution to the adverse effects caused by the lack of synchronization in heterogeneous TSN networks that combine one or more Ethernet-based legacy systems through a TSN communications subsystem. On the contrary, most of the conducted studies aim to integrate TSN with both wireless and 5G networks. For example, a low-overhead beacon-based time synchronization method was implemented to provide precise synchronization in wireless networks in the context of highly deterministic TSN networks, as outlined in [10]. Other research has focused on extending IEEE 802.1AS and IEEE 802.11 to enable TSN integration with wireless networks, as described in [4] and [11]. Additionally, the challenges of integrating Wired TSN and WLAN technologies and a possible solution in the form of a hybrid TSN device architecture were discussed in [14]. Moreover, the study in [9] presented TSN clock synchronization that aligns with 5G specifications. To solve cross-domain clock synchronization issues in 5G-TSN networks, a method based on data packet relay was proposed in [7]. Finally, the performance of 5G-TSN networks was also evaluated in terms of clock synchronization in several works such as in [16, 15], and in [17].

On the other hand, one methodology for integrating EtherCAT and TSN in terms of clock synchronization is presented in [5]. However, this type of integration requires customized solutions for each protocol being integrated, which can pose a challenge to the wider adoption of TSN by the industry. This is because designing and implementing these solutions take significant time

and resources, and compatibility between solutions can also be demanding.

In a work presented in [6], the authors implemented a non-TSN network with its own synchronization protocols and they replaced its communications subsystem with TSN. The work preliminary identified the effects of lack of synchronization between the legacy system and the TSN network due to the lack of integration between the synchronization protocols used by the legacy system and the TSN's Generalized Precision Time Protocol (gPTP) [2]. Through several experiments, authors detected the causes and consequences of the lack of synchronization in the short and long term in the network. However, the work was a short paper that merely suggested uncertain and indeterminate solutions that lacked implementation and proper validation.

11.3 Background

In TSN networks, communication between end-stations is achieved by the transmission of Ethernet frames along Ethernet links and TSN switches. In both TSN switches and end-stations, each output port has up to 8 FIFO queues, each corresponding to one specific priority level. TSN frames are assigned to one of the 8 priorities, or queues, which are configured as one of the 3 types of TSN traffic including TT, Audio Video Bridging (AVB), and Best-Effort (BE) traffic. TT traffic is commonly given the highest priority, while BE traffic has the lowest priority. Several queues can be configured as the same type of traffic thus giving different classes, for example, AVB class A, B, and C. An illustration of these concepts can be seen in Fig. 11.1, which shows a TSN device (either an end-station or switch) output port with four queues configured to convey two TT traffic classes with the highest priority, one AVB traffic class with medium priority, and BE traffic with the lowest priority.

Next, we explain 3 key aspects of the background for this work. First, we will introduce the Time Aware Shaper (TAS) and the gPTP, since they are the main mechanisms responsible for TT transmission and the most affected by the lack of synchronization. On the other hand, we will explain the Centralized Network Configuration element (CNC), a key component for TALESS implementation.

11.3.1 Time Aware Shaper

To provide the determinism required by TT traffic and, therefore, to know exactly when each TT frame is transmitted, TSN must be able to prevent inter-frame interference. To do this, TSN uses the TAS mechanism shown in Fig. 11.1. This mechanism assigns a gate to each queue that can be open

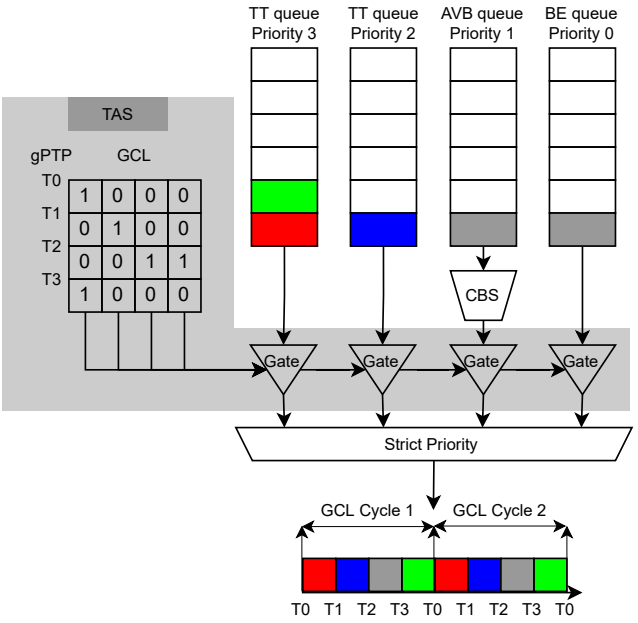


Figure 11.1: A TSN egress port with four FIFO queues: two TT queues, one AVB queue, and one BE queue.

or closed. The state of the gate is determined by the Gate Control List (GCL) which specifies at the nanosecond level how long a gate should be open or closed in a cyclically repeating list. If the gate of a queue is open, it can transmit the traffic in the queue, otherwise, the frames in that queue are blocked from transmission. The opening period of a gate is called a *transmission window* or simply a *window*.

The operation of TAS for two TT queues is also depicted in Fig. 11.1. In this example, three TT frames with a period of 4 time units and transmission time of 1 time unit are transmitted through a TSN switch port, where two of the frames are assigned the highest priority 3 (green and red) and one frame (blue) is assigned priority 2. In order to schedule the transmissions, the hyper-period, which is the least common multiple of the frames' periods, is calculated. This value is used to define the cycle of the GCL, which is responsible for controlling the transmission of the frames by specifying the open or closed state of the gates associated with each priority queue. Thus, the GCL cycle in this example is set to 4 time units, hence the list will be repeated every 4 time units. From time T0 to T1, the gate for priority 3 queue is open, allowing the transmission of the red frame, while the gate for the other queues remains closed. From

T1 to T2, the blue frame, which has priority 2, can be transmitted as its gate is open. Both gates are closed between T2 and T3, resulting in no TT transmission but allowing lower priority queues to transmit even if higher priority frames are waiting for transmission. Finally, the gate for the priority 3 queue is open in the last transmission window, allowing the transmission of the green frame. The bottom of Fig. 11.1 displays two cycles of frame transmissions which shows repetition of the GCL list.

11.3.2 Generalized Precision Time Protocol

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 (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 TSN devices. The PDM mechanism is used once the hierarchy is established in order to measure the propagation delay between systems. Finally, the TTI mechanism is used to forward the grandmaster time which, together with the measured propagation delay, is used to synchronize the other TSN devices.

This synchronization protocol can achieve a clock accuracy of tens of nanoseconds. However, it has stringent software and especially hardware requirements that in most cases legacy devices from Ethernet-based networks cannot support.

11.3.3 Centralized Network Configuration element

The CNC is a virtual component that can be placed in a designated node, an end-station, or a switch. Regardless of its placement, it can exchange information with network devices via NETCONF [8, 1]. This communication is bidirectional, allowing end-stations to send user or network configuration requests to the CNC while switches can communicate their specifications. Finally, the CNC can distribute new configurations to the entire network.

NETCONF utilizes a client-server approach for configuring the network, where the CNC acts as the client, responsible for collecting network information and initiating network device configurations. Note that all TSN network devices, e.g, TSN switches, must have a NETCONF server enabled in order to receive configurations from the CNC.

Figure 11.2: Heterogeneous TSN network with legacy end-stations topology.

11.4 Problem statement

In order to observe the problems caused by the lack of synchronization between legacy systems and the TSN communication subsystems, we set up a small legacy network consisting of two single-board computers, i.e., Raspberry Pi (RPi) 3 Model B, running RPi Operating System (OS) and connected point-to-point. Afterward, we add a Multiport TSN kit switch from the company System-on-Chip Engineering (SoC-e)[1] so that the RPIs behave as legacy end-stations in the new network, see Fig. 11.2. The Raspberry Pi boards are configured to synchronize their software clocks with each other via the Network Time Protocol (NTP). Note that any clock synchronization protocol other than gPTP could be used between the legacy end-stations since they were reproducing scenarios where the TSN switch is unable to synchronize with the end-stations.

In this experiment, we start analyzing the legacy network separately, i.e., without the TSN network, to see its baseline behavior. Then, a TSN network is added to the legacy system to analyze its effects. Through these experiments, it can be observed that thanks to the improved hardware and software capabilities of the TSN switches, the jitter of the legacy network practically disappears. However, due to the lack of synchronization, there is a drift between the clock time of TSN and the legacy system. This causes a deviation between the communication schedule of the legacy system and the TSN schedule that can be either positive or negative depending on which clock is faster or slower. Below we explain the findings of the experiment in detail.

Fig. 11.3 shows the behavior of a heterogeneous TSN network, in which the legacy system experiences a positive clock drift relative to the TSN communication subsystem. When the TSN clock is slower than the legacy system clock, the legacy system schedule exhibits a positive drift relative to the TSN schedule, causing frames to arrive at the receiver increasingly later than their

[1]MTSN Kit: a Comprehensive Multiport TSN Setup. [Online]. Available: `https://soc-e.com/mtsn-kit-acomprehensive-multiport-tsn-setup/`

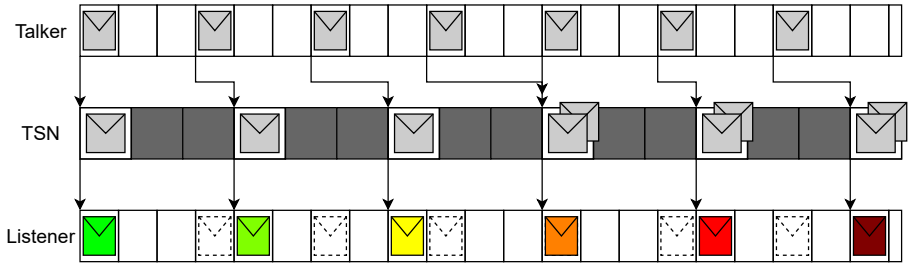


Figure 11.3: Positive legacy system clock drift behavior.

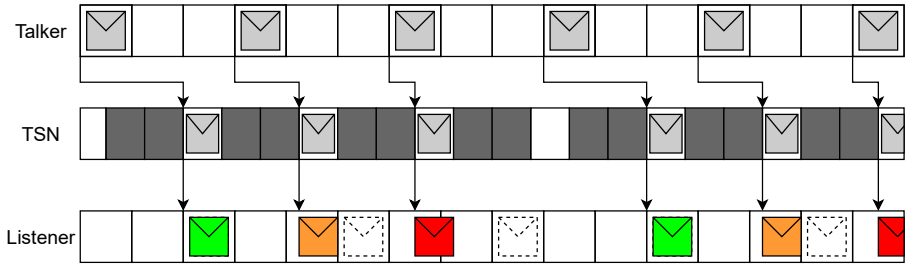


Figure 11.4: Negative legacy system clock drift behavior.

legacy scheduled time. Moreover, since the transmission of frames by the TSN network to the legacy system receiver (*listener* in TSN terminology) is slower than the transmission by the legacy system transmitter (*talker* in TSN terminology) to the TSN network, the frames stack up in the buffers. However, the buffers are not infinite, hence frames that arrive once the buffer is full are discarded.

Fig. 11.4 shows the behavior of a heterogeneous TSN network, in which the legacy system experiences a negative clock drift relative to the TSN communication subsystem. When the TSN clock is faster than the legacy system clock, the legacy system schedule exhibits a positive drift relative to the TSN schedule, causing frames to arrive at the receiver increasingly earlier than their legacy scheduled time. However, this effect cannot be infinitely extended over time since it is impossible to receive a frame before it has been transmitted. When enough drift accumulates after a while, frames miss the transmission window in which they are scheduled, leaving a period with no frames being transmitted.

Through these experiments, we can observe that legacy systems can continue communicating through TSN and even benefit from some of its features such as improved reception jitter. However, due to the lack of synchronization, a clock drift appears which not only causes a deviation in reception but

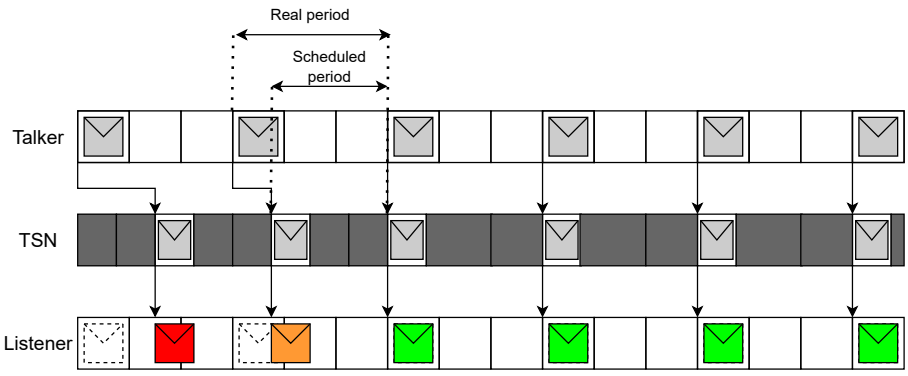


Figure 11.5: TALESS operating diagram.

can lead to empty transmission windows or even loss of frames. Therefore, the objective of this work is to develop a mechanism that eliminates the drift between the TSN schedule and the legacy system schedule without requiring any modification in the legacy end-stations.

11.5 TALESS: TSN with Legacy End-Stations Synchronization

As it is described in the previous section, drift is the main cause of errors when having a lack of synchronization. However, there are two factors to consider when developing a solution. First, different legacy systems that are not synchronized with each other may have different time domains, i.e., different drifts with respect to the TSN network. Secondly, such a drift may not be constant over time as environmental factors, such as temperature, may differently affect the different clocks in the heterogeneous network. Therefore, the proposed solution should eliminate the effects of the clock drift of different legacy systems that changes over time.

One way to avoid the negative consequences of the drift caused by the lack of synchronization consists in eliminating the drift between the TSN network schedule and the legacy system. In order to achieve this, we propose to modify the size of the TSN GCL transmission windows when there is drift. This way, we can modify the TSN's transmission pace to match the legacy system's. Fig. 11.5 shows an example of the operation of the proposed solution. This figure shows how, after detecting the drift, the TSN network changes the size of certain windows in a way that from that point onward the frames arrive to the receiver according to the legacy system schedule. However, the TT traffic

transmission windows should not be modified since the size of these windows is determined by the size of the frame and the transmission rate, where both parameters are independent of the clock drift. In this regard, if a network reaches 100% utilization and the clock of the legacy system becomes faster with respect to the TSN network, it would not be possible to implement the solution since there would not be sufficient resources in the TSN network. However, configuring to 100% utilization on the network is impractical and industrial use cases commonly avoid that. Therefore, in TALESS, non-TT windows (NTTW) should be modified by a ratio equal to the drift between the legacy system and TSN (D) plus the cumulative variation in TT windows (TTW). Therefore, the new size of each NTTW ($NTTW_i.size'$) can be computed as:

$$\begin{aligned} NTTW_i.size' = NTTW_i.size \ + \ D \times NTTW_i.size + \\ D \times (NTTW_i.start \ - \ NTTW_{i-1}.end) \end{aligned} \tag{11.1}$$

where, to the previous NTTW size $NTTW_i.size$, we first add the variation of the window by multiplying the previous size $NTTW_i.size$ by the drift percentage D (either positive or negative), and secondly we add the cumulative variation of the TTW between the previous NTTW $NTTW_{i-1}$ and the current one. This last increment is due to the fact that, as TTW cannot be modified, the increment of these windows accumulates until the next NTTW.Revisiting the Fig. 11.5, we can observe that the implementation of the Eq. (11.1) results in the expansion of the gray transmission windows, which correspond to the NTTWs, allowing them to match the transmission pace of the legacy system. Moreover, the NTTWs located after a TTW exhibit a longer extension due to their assimilation of the expansion that corresponds to the TTW.

Eq. (11.1) would be sufficient in a heterogeneous network where the drift between the legacy system and the TSN network is constant. In that case, it would be enough to calculate the drift and apply the formula to the TSN schedule only once offline. However, if the drift is variable or if several legacy systems with different time domains coexist in the same TSN network, the previous solution will not be sufficient. Regarding variable drifts, constant monitoring and reconfiguration of the network is necessary. To do this, we propose a Drift Detector (DD) that detects the drift between different clocks continuously during run-time. Thus, we also propose to implement a reconfiguration mechanism in the CNC, as shown in Fig. 11.6. The DD, which is located on at least one reception port of a switch connected to a legacy system talker, samples the reception and by comparing the clock with the TSN schedule determines the drift between the legacy system and the TSN network. When it exceeds an established threshold, which can be set by a user, a signal is sent to the CNC informing about the drift value. The CNC then updates the network

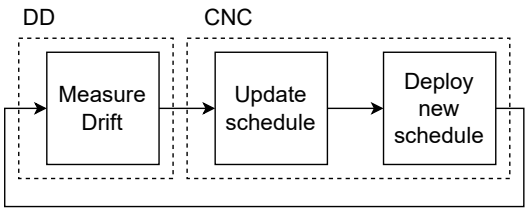


Figure 11.6: TALESS task flow.

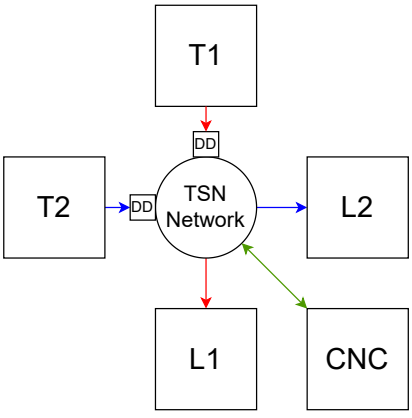


Figure 11.7: TALESS architecture.

configuration according to Eq. (11.1) and deploys it on the network to eliminate the drift. The diagram in Fig. 11.7 depicts a network that implements TALESS, in which end-stations T1 and T2, as well as L1 and L2, represent the Talkers and Listeners of legacy systems 1 and 2, respectively.

Finally, to allow the solution to work in networks combining different legacy systems, the only requirement is that TT traffic routes of different legacy systems cannot share output ports. This is because variations between the drifts of the legacy systems would invalidate TSN scheduling since the different drifts could cause some transmission windows to be advanced while others are delayed, causing them to collide. Moreover, given the small variability of the clocks, the resulting hyper-periods would be exponentially long. For example, if two legacy systems transmit with 1 second period but one has a 1% positive drift and the other one has 1% negative, instead of a GCL of 1 second with 3 transmission windows, the GCL would have an extension of lcm(1.01,0.99)=99.99 seconds with more than 200 transmission windows.

11.6 TALESS Validation Setup

In this paper, we verify the effectiveness of the solution using two methods: a simulation model of the solution at the end-stations and an experimental implementation.

11.6.1 Simulation Model

Our model simulates the behavior of a TSN switch implementing TALESS. However, since TALESS solely eliminates drift, the results obtained in our experiments can be extrapolated to larger networks with any type of schedule, as long as the network architecture and schedule are functional in the absence of drift.

The model is implemented in Matlab and uses a number of parameters as inputs. These parameters include the period of the transmission to be modeled, the cumulative drift over the modeled run-time, and the jitter of the received transmission as the variance of a specified distribution. In addition, the modeled network run-time must be specified as an input. This is one of the main advantages of the model over the experimental implementation since, as real drifts are very small, the effects are noticeable only in the long term. In this sense, the model allows us to analyze long periods of time with realistic drift values in a reasonable model execution time.

The reception of frames is modeled as a list of timestamps (ts) generated by applying the drift variation (dv) and jitter (j) to the period (p), i.e.

$$ts_i = ts_{i-1} + p * dv^i + normrnd(0, var) \tag{11.2}$$

where $normrnd(0, var)$ is a random value following a specific distribution, in this case, a normal distribution, with mean 0 and the variance var corresponding to the variance of the jitter used as an input. All ts values in the list are analyzed one by one by the DD module. The DD module determines whether the period of the reception is equal to the initially scheduled one by means of a Student's t-test (`ttest`[2]). Once a significant difference is detected, i.e. the probability of the periods being equal is below a predetermined threshold, the period is updated based on the trend measured in the frames received since the last period update (`polyfit`[3]).

[2]One-sample and paired-sample t-test - MATLAB ttest [Online]. Available: `https://se.mathworks.com/help/stats/ttest.html`

[3]Polynomial curve fitting - MATLAB polyfit - MathWorks [Online]. Available: `https://se.mathworks.com/help/matlab/ref/polyfit.html`

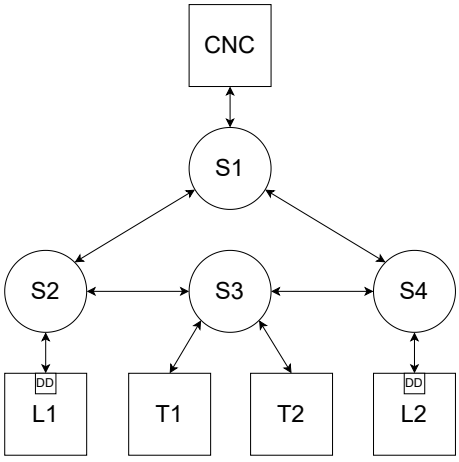


Figure 11.8: Experimental network diagram showing TSN Switches (S) and legacy systems 1 and 2 represented by Talkers (T) and Listeners (L).

Finally, the model shows three different results for both positive and negative drift. The first result is the behavior of the reception with free transmission, i.e., without the intervention of the TSN switch, while the second result is the behavior with a fixed schedule without applying any solution. Finally, the last result is the effect of TALESS implementation. The results will be presented and discussed in Section 11.7.

11.6.2 Experimental Setup

For the experimental implementation, we extended the network presented in Section 11.4. More specifically, we use 4 Raspberry PIs and 4 TSN switches, and a computer that will act as a CNC. The architecture of the new network is illustrated in Fig. 11.8, where T1 and L1 represent the talker and listener of legacy system 1, and T2 and L2 the ones of legacy system 2. In addition, S1 to S4 and the CNC represent the TSN communications subsystem.

Each pair of Raspberry PIs (Ti, Li) forms an independent legacy system, i.e., they are not synchronized nor communicate with the end-stations of the other legacy system. For each talker, we implemented a synthetic clock with different drift values with respect to TSN communications subsystem that change throughout the experiment. These drift values were larger than those present in a normal network in order to magnify the effects in a reasonable duration of the experiments. In addition, in one of the legacy systems, the drift grew positively, while in the other it grew negatively. In each legacy system, its synthetic clock is responsible for driving the transmission. In order to keep

the talker and the listener synchronized, apart from the previously mentioned NTP, every time the synthetic clock drift changes, the talker sends a message to the listener with the new synthetic clock frequency value so that the listener can update its own synthetic clock.

According to the design, sketched in Section 11.5, the DD should be implemented in the input port of switches to avoid modifications in the legacy end-stations. However, since we do not have access to the implementation of switches, we implemented the DDs in the legacy listener. Despite the change of the DDs location, neither the calculation method nor the obtained drift value changes. This is because the DDs are capable of monitoring the drifts on the ports, either connected to switches or to the legacy end-stations. Once the DD measures a significant clock difference, it sends the drift value to the CNC.

The CNC is based on the implementation proposed in [3], which was openly available to the research community. It uses a JSON file with the configuration to be deployed in the TSN network and NETCONF to deploy the configuration. The CNC is implemented to receive drift information from the DD, update the configuration based on Eq. (11.1), and automatically deploy the improved configuration in the TSN network. The results are presented and discussed in Section 11.7.

11.7 Simulation and experimental results

In this section, we will show and analyze the results obtained using the simulation model and TALESS experimental implementation. In addition, we will compare the model with the experimental implementation to validate both the proposed solution and the simulation model.

To analyze the obtained results, we will use a metric called Synchronization Quality Metric (SQM). This is calculated by dividing the difference between the Reception Time (RT) of two consecutive frames minus the Scheduled Period (SP) for those frames by the SP, i.e.,

$$SQM_i = \frac{(RT_{i+1} - RT_i) - SP}{SP}. \tag{11.3}$$

The SQM allows us to analyze drift and jitter graphically since the mean SQM in a given interval provides information about the drift of the receptions in such intervals while the maximum absolute value of SQM minus the mean SQM provides the ratio of jitter with respect to the period. Note that this metric does not allow us to observe extreme cases such as frame loss, since the SQM cannot be quantified due to the lack of RT.

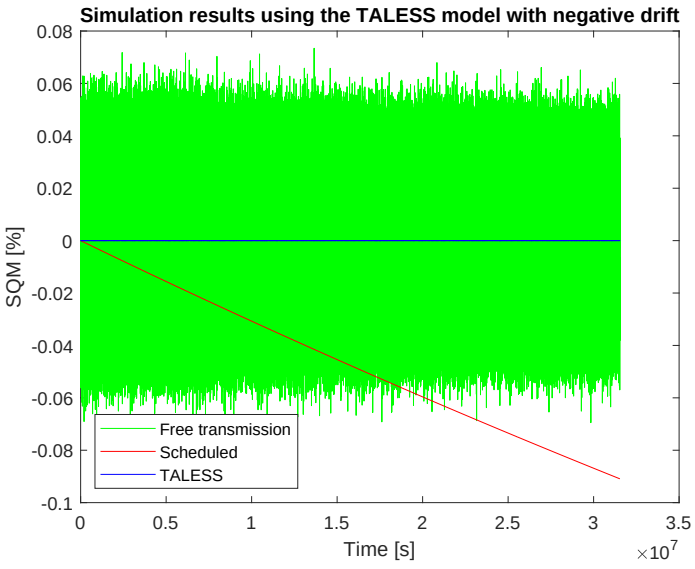


Figure 11.9: Simulation results of one year of transmissions in a heterogeneous TSN network with negative clock drift in three different scenarios: free, scheduled, and TALESS transmission.

Finally, all the analyses will be performed by comparing the reception of periodic frames in three different scenarios: (i) with free traffic flow through the TSN network, i.e., without applying TAS or any other scheduling mechanism, (ii) with the TSN communication subsystem scheduled without TALESS implementation, and (iii) with TALESS implementation.

11.7.1 Simulation Model Results

We simulate two different scenarios using the simulation model. In both cases, the model simulates a year of communications of a periodic transmission with an initial period of 1 second, and with a variable drift that starts at 0% and grows progressively until reaching 10% at the end of the simulation in the first scenario and from 0 to -10% in the second one. In addition, the jitter of the transmission is used as an input to the model and follows a normal distribution of variance 0.01. This distribution and variance are similar to the ones in Section 11.4. The results can be seen in Figs. 11.9 and 11.10, both showing the SQM over the simulation time.

In both scenarios, we observe that the free reception has a jitter of 70 ms (as defined as input) and zero drift. When scheduling the TSN subsystem with-

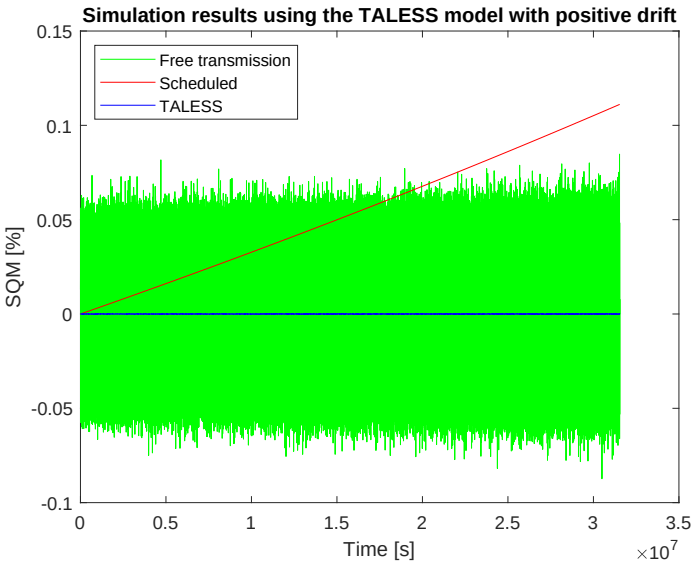


Figure 11.10: Simulation results of one year of transmissions in a heterogeneous TSN network with positive clock drift in three different scenarios: free, scheduled, and TALESS transmission.

out TALESS, the jitter almost disappears but the effects of the drift between the TSN schedule and the legacy transmission become evident. Finally, we observe that by applying TALESS both the jitter and the drift almost drops to 0.

11.7.2 Real Network Implementation Results

Using the real network we run an experiment similar to the model but with certain restrictions. Instead of a year of execution, only 2000 frames are transmitted per experiment and the drift, instead of increasing and decreasing progressively up to $\pm 10\%$, varies by $\pm 5\%$ every 100 frames. Moreover, in the positive drift scenario, the legacy system starts with a period of 1 second that is periodically shortened, while in the negative drift scenario, it is the final period which is equal to 1 second. All other characteristics and scenarios are the same as in the simulation model. The results of these experiments can be seen in Figs. 11.11 and 11.12.

As in the model, here we can see how the free transmission has high jitter and no drift. Once the scheduling is applied without TALESS, the jitter disappears but the drift takes place. In this case, the SQM (and therefore the drift)

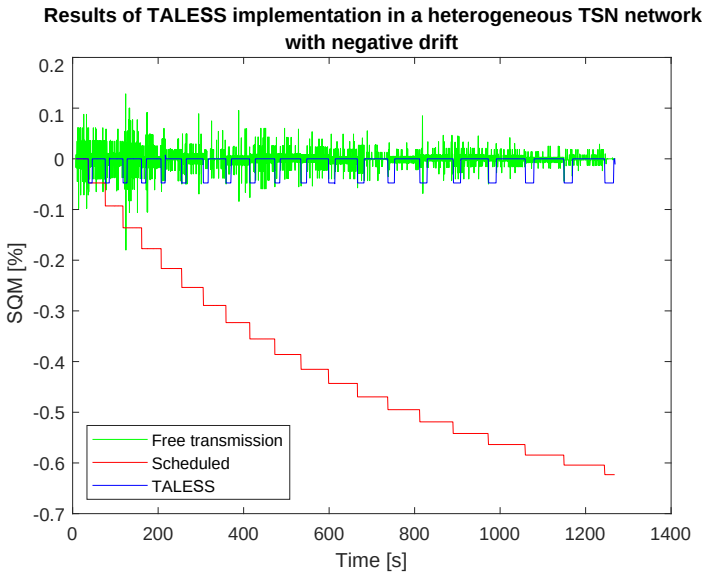


Figure 11.11: Results of heterogeneous TSN network execution with negative drift in three different scenarios: free, scheduled, and TALESS transmission.

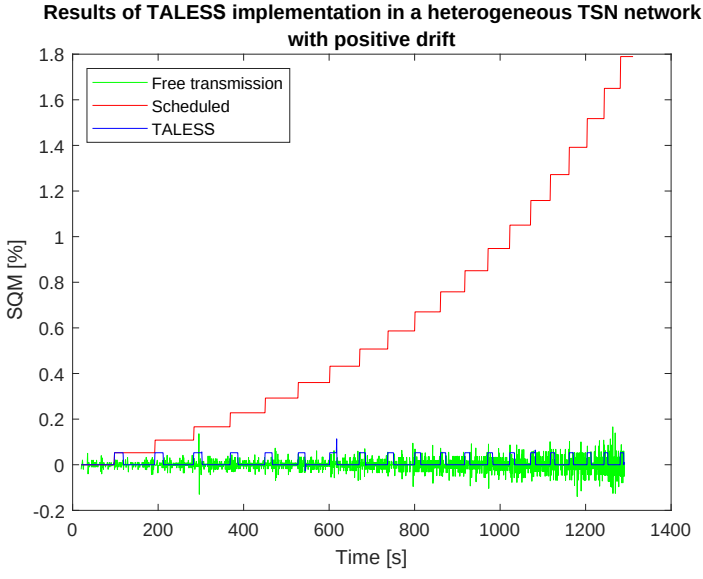


Figure 11.12: Results of heterogeneous TSN network execution with positive drift in three different scenarios: free, scheduled, and TALESS transmission.

presents a step-wise behavior instead of a continuous one because, as previously mentioned, the drift variation is applied every 100 frames for the sake of simplicity in the experiment.

Finally, we see how TALESS eliminates both jitter and drift yet leaves some drift remnants (the duty cycle that is observed in the figures). These are due to the time required by the solution to detect the change in the reception and are larger than what is observed in the simulation model due to the large synthetic drift applied to this experiment to allow us to visualize the effects of TALESS on the drift in a reasonable time. Although small periodic drifts can accumulate significant drift times between the TSN network and the legacy system, there are ways to prevent this, e.g., by over-correcting the drift by creating equivalent drifts but of opposite sign to ensure an overall average drift equal to 0.

11.7.3 Comparison Results

Finally, in order to validate both the simulation model and the experimental implementation, we modified the model to run a simulation with the same conditions applied to the experimental implementation, i.e., execution of only 2000 frames with a variable drift of $\pm 5\%$ every 100 frames. The result of such simulations is shown in Figs. 11.13 and 11.14.

As we can see, the simulations of implemented scenarios match the implementation results. Although the transmission by the legacy talker is not exactly the same since the real network does not exactly follow a normal distribution, the effects of both schedulings (with and without TALESS) on reception are essentially the same. This experiment provides evidence that the simulation model follows the experimental results ensuring the validity of the simulation model and, therefore, of TALESS.

11.8 Conclusions and Future work

In this paper, we analyzed the effects of the lack of synchronization between the legacy systems with the TSN network. These effects are mainly due to the drift between clocks in TSN and in the legacy systems which, in the long term, will result in either delayed TT transmission or missing frames. Therefore, we designed, implemented, and validated a solution, called TALESS, to remove the identified effects. Through simulation and implementation of TALESS, we demonstrated that TALESS efficiently enforces the reduction of jitter and removes the effects of clock drifts in the legacy systems. This solution allows us

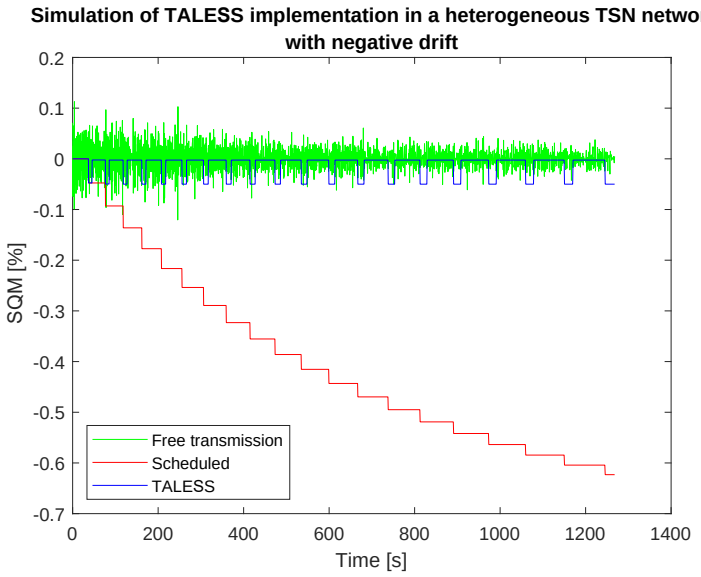


Figure 11.13: Simulation results of the implemented heterogeneous TSN network with negative drift.

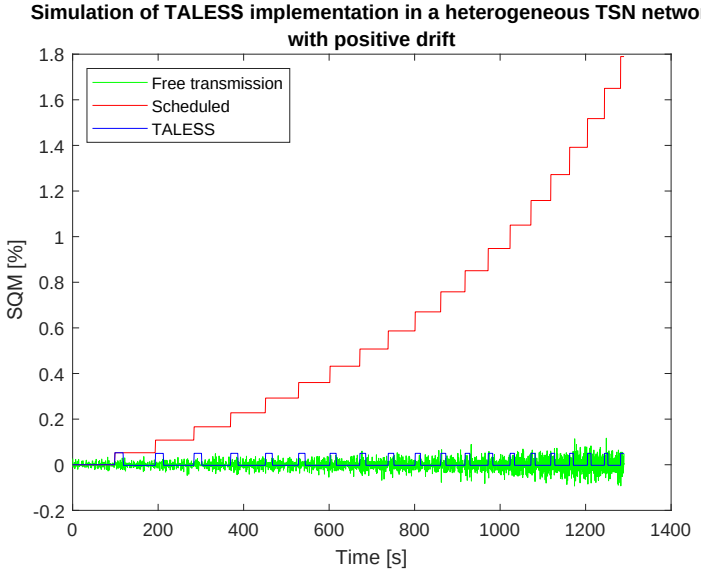


Figure 11.14: Simulation results of the implemented heterogeneous TSN network with positive drift.

to integrate several legacy systems into a TSN network without any modifications in their clock synchronization.

As the future work, we aim to implement the proposed mechanism within a TSN switch to provide a complete tool for TSN adoption without any modification within the legacy systems.

Bibliography

[1] IEEE 802.1Qcc-2018 - IEEE Standard for Local and Metropolitan Area Networks - Bridges and Bridged Networks - Amendment 17: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements, 2018. DOI: 10.1109/IEEESTD.2018.8372875.

[2] IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications. *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pages 1–421, 2020.

[3] Inés Alvarez, Andreu Servera, Julián Proenza, Mohammad Ashjaei, and Saad Mubeen. Implementing a First CNC for Scheduling and Configuring TSN Networks. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, 2022.

[4] Haytham Baniabdelghany, Roman Obermaisser, et al. Extended synchronization protocol based on IEEE802. 1AS for improved precision in dynamic and asymmetric TSN hybrid networks. In *IEEE Mediterranean Conference on Embedded Computing (MECO)*, pages 1–8, 2020.

[5] Daniel Bujosa, Daniel Hallmans, Mohammad Ashjaei, Alessandro V Papadopoulos, Julian Proenza, and Thomas Nolte. Clock synchronization in integrated tsn-ethercat networks. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 214–221, 2020.

[6] Daniel Bujosa, Andreas Johansson, Mohammad Ashjaei, Alessandro V Papadopoulos, Julian Proenza, and Thomas Nolte. The Effects of Clock Synchronization in TSN Networks with Legacy End-Stations. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, 2022.

[7] Zichao Chai, Wei Liu, Mao Li, and Jing Lei. Cross Domain Clock Synchronization Based on Data Packet Relay in 5G-TSN Integrated Network. In *IEEE International Conference on Electronics and Communication Engineering (ICECE)*, pages 141–145, 2021.

[8] Rob Enns. RFC 4741: NETCONF Configuration Protocol, 2006.

[9] Michael Gundall, Christopher Huber, Peter Rost, Rüdiger Halfmann, and Hans D Schotten. Integration of 5G with TSN as prerequisite for a highly flexible future industrial automation: Time synchronization based

on IEEE 802.1 AS. In *IECON Annual Conference of the IEEE Industrial Electronics Society*, pages 3823–3830. IEEE, 2020.

[10] Jetmir Haxhibeqiri, Xianjun Jiao, Muhammad Aslam, Ingrid Moerman, and Jeroen Hoebeke. Enabling TSN over IEEE 802.11: Low-overhead time synchronization for wi-fi clients. In *IEEE international conference on industrial technology (ICIT)*, volume 1, pages 1068–1073, 2021.

[11] Alexey M Romanov, Francesco Gringoli, and Axel Sikora. A precise synchronization method for future wireless TSN networks. *IEEE Transactions on Industrial Informatics*, 17(5):3682–3692, 2020.

[12] R. Salazar, T. Godfrey, L. Winkel, N. Finn, C. Powell, B. Rolfe, and M. Seewald. Utility Applications of Time Sensitive Networking White Paper (D3). Technical report, IEEE, 2018.

[13] S. Samii and H. Zinner. Level 5 by Layer 2: Time-Sensitive Networking for Autonomous Vehicles. *IEEE Communications Standards Magazine*, 2(2):62–68, 2018.

[14] Oscar Seijo, Xabier Iturbe, and Inaki Val. Tackling the Challenges of the Integration of Wired and Wireless TSN with a Technology Proof-of-Concept. *IEEE Transactions on Industrial Informatics*, 2021.

[15] Haochuan Shi, Adnan Aijaz, and Nan Jiang. Evaluating the Performance of Over-the-Air Time Synchronization for 5G and TSN Integration. In *IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pages 1–6, 2021.

[16] Jiajia Song, Makoto Kubomi, Jeffrey Zhao, and Daisuke Takita. Time synchronization performance analysis considering the frequency offset inside 5G-TSN network. In *IEEE International Symposium on Wireless Communication Systems (ISWCS)*, pages 1–6, 2021.

[17] Tobias Striffler and Hans D Schotten. The 5G Transparent Clock: Synchronization Errors in Integrated 5G-TSN Industrial Networks. In *IEEE International Conference on Industrial Informatics (INDIN)*, pages 1–6, 2021.

[18] M. Wollschlaeger, T. Sauter, and J. Jasperneite. The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0. *IEEE Industrial Electronics Magazine*, 11(1):17–27, 2017.

Chapter 12

Paper E
CSRP: An Enhanced Protocol for Consistent Reservation of Resources in AVB/TSN.

Daniel Bujosa, Inés Álvarez, Julián Proenza.
In the IEEE Transactions on Industrial Informatics 17 (TII 2020).

Abstract

The IEEE Audio Video Bridging (AVB) Task Group (TG) was created to provide Ethernet with soft real-time guarantees. Later on, the TG was renamed to Time-Sensitive Networking (TSN) and its scope broadened to support hard real-time and critical applications. The Stream Reservation Protocol (SRP) is a key work of the TGs as it allows reserving resources in the network, guaranteeing the required quality of service (QoS). AVB's SRP is based on a distributed architecture, while TSN's is based on centralized ones. The distributed version of SRP is supported and used in TSN. Nevertheless, it was not designed to provide properties that are important for critical applications. In this work we model SRP using UPPAAL and we study the termination and consistency. We verify that SRP does not provide such properties. Furthermore, we propose an improved protocol called Consistent Stream Reservation Protocol (CSRP) and we formally verify its correctness using UPPAAL.

12.1 Introduction

The IEEE Audio Video Bridging (AVB) Task Group (TG) [9] was created in 2005. Its purpose was creating a set of standards to provide Ethernet with soft real-time capabilities oriented to applications related to audio/video streaming. Specifically, the AVB TG started three projects, namely the IEEE Std 802.1AS [4], dedicated to clock synchronization; the IEEE Std 802.1Qav, which standardized the Credit-Based Shaper [2]; and, finally, the IEEE Std 802.1Qat, which standardized the Stream Reservation Protocol (SRP) [3]. In addition, the TG created a profile that sets a series of rules to ensure a minimum Quality of Service (QoS) when using the aforementioned standards. The profile is the IEEE Std 802.1BA-2011: Audio Video Bridging Systems [5]. This set of standards is commonly referred to as AVB standards.

Over time, the interest in the work done by the TG grew, also in areas of application beyond audio/video streaming, such as automotive [27], automation [30] and energy distribution [26]. For this reason, in 2012 the group was renamed to Time-Sensitive Networking (TSN) TG and its target broadened to meet the needs of these new applications, which are usually based on Critical Distributed Embedded Systems (CDES). Specifically, the TSN TG aims at providing Ethernet with proper support for mixed hard and soft real-time communications, flexibility of the traffic requirements and fault tolerance mechanisms. The set of standards developed by the TG is usually referred to as TSN standards.

Although the number of standardization projects carried out by the TSN TG grows at high speed, there are some projects that can be considered the core of the TG activity. One of the key projects is SRP, which was originally standardized by the AVB TG in [3] and subsequently reviewed by the TSN TG in [7]. SRP allows Ethernet to reserve resources along the path that connects a transmitter to one or more receivers. More concretely, SRP only authorizes the transmission of messages after verifying that the network can convey such messages with the required QoS. This prevents frame delays over the predefined limits during transmission and frame losses due to overflows of buffers in bridges. Moreover, SRP allows modifying the traffic requirements at run-time, providing a certain degree of flexibility to the network.

Currently, there are three different SRP architectures defined by the TSN TG in [7]. The first one is the fully distributed architecture, created in the context of AVB; whereas the other two architectures are centralized and they were defined in the context of TSN. TSN relies on YANG [12] to configure the network when using centralized architectures. YANG is a data modeling language which allows to define which data must be used to configure a network device

and which is the format of said data. This allows to standardize and simplify the integration of different processes and applications in distributed systems.

We must note that, at the moment of writing this paper, there is no available YANG model to support the online configuration of the reservations of event-triggered traffic using the centralized architectures proposed in TSN [8]. Therefore, the distributed version of SRP is still used in TSN to manage the network resources of event-triggered real-time traffic. For this reason, and for the interest that certain areas such as automotive have shown on the distributed SRP [29, 20, 21, 18, 25, 28, 15], we believe that this version of SRP is going to continue to be used in the upcoming years.

As we have said, TSN targets critical applications, which means that all protocols, including every version of SRP, must exhibit certain properties which are common in CDES to ensure the proper behavior of the overall system. In this work we focus on termination and consistency. On the one hand, it is common for applications executed by CDES to carry certain actions within a bounded time and, thus, termination must be guaranteed. On the other hand, nodes in CDES usually need to have a consistent view of the network or share data consistently to interact with each other correctly.

Nevertheless, the distributed version of SRP was not developed to be used in CDESs. Even though we can see that the distributed SRP does not guarantee termination nor consistency with a simple analysis, it is not effective to thoroughly identify the potential scenarios without using any formal tool. For this reason, in this work we use a formal model checker to verify in an exhaustive manner whether the distributed version of SRP provides these properties and in which cases it does not. From now on, whenever we say SRP we refer to the distributed version of the protocol.

Specifically, we use the UPPAAL model checker [11] to build a model of SRP. Modeling any system or protocol requires to abstract certain details, as analyzing all possible scenarios in an exhaustive manner requires a great amount of memory and time. For this reason, our SRP model abstracts implementation details of the protocol. Nevertheless, the level of abstraction used in this work is the typical when modeling communication networks. Moreover, we validate our model, understanding the term validate as the evaluation done to ensure that the model is properly implemented and that it is a faithful representation of the behavior of SRP.

We then use our UPPAAL model to verify that SRP does not provide termination nor consistency and to detect in which scenarios this happens. We use the term verify to refer to the evaluation done to ensure that a system presents a certain property, i.e., our system is the right one for our needs. Moreover, we discuss the consequences derived from the absence of termination and consis-

tency. We propose different ways to modify SRP in order to provide it with the aforementioned properties and we select what we consider to be the best one. Finally, we create a UPPAAL model of the modified protocol, which we call Consistent Stream Reservation Protocol (CSRP). Again, we validate our model and we verify the correctness of our design.

The remainder of the document is structured as follows. Section 12.2 summarizes the related work, while Section 12.3 explains the parts of SRP that are most relevant for this work. Section 12.4 provides an overview of the SRP model we implemented in UPPAAL while Section 12.5 and Section 12.6 show the termination and consistency issues detected and their consequences. Section 12.7 describes the solution proposed. Section 12.8 describes the changes applied to the SRP model to implement the proposed CSRP while Section 12.9 and Section 12.10 show the formal verification of the correctness of CSRP. Finally, Section 12.11 summarizes the work done.

12.2 Related Work

Due to the great relevance of the AVB and TSN standards, the community has carried out a significant amount of work related to their study, application and improvement. For example, in [16] authors describe the requirements for an AVB network, summarize the methods described in the standards and describe how they can be used by several higher layer protocols; while in [23] authors provide an up-to-date comprehensive survey of the TSN standards and the related research studies.

Furthermore, there are many works related to the study of AVB's efficiency, such as [21, 25, 19, 22]. On the other hand, in the work presented in [24] the authors detect a drawback in the resource reservation de-registration specification, which leads to the waste of the network resources, and proposed some solutions. Moreover, some works present solutions to provide fault tolerance against permanent faults using SRP [17].

Nevertheless, to the best of the authors' knowledge, there are no works related to the study of the termination and consistency of the distributed version of SRP, apart from the preliminary work presented in [13]. We next list the contributions of this paper compared to the work presented in [13]. In this work we carry out an improvement of the UPPAAL model of SRP resource reservation mechanism which is explained in more detail in Section 12.4. We study the termination and consistency of the reservations in different scenarios and components with this new UPPAAL model. We design an enhanced version of SRP that does exhibit the termination and consistency properties, thereby eliminating all the issues we have identified for SRP and that provides

the network devices with enough information to make rather complex decisions about the reservation of resources within a bounded time. Finally, we implement the proposed solution, i.e., CSRP, in the UPPAAL model and we verify that it is correct.

12.3 SRP Overview

As we anticipate in Section 12.1, SRP is a key piece for many of the projects developed by the AVB and TSN TGs. Specifically, SRP is key to provide real-time guarantees to Ethernet-based communications. More concretely, SRP allows verifying that there are enough resources in the network to convey the traffic and reserving said resources. This allows to bound the end-to-end delay of the frames and to avoid the loss of packets due to the buffer overflow. Moreover, SRP can be used to modify the traffic requirements at run-time, giving the network a certain degree of flexibility.

SRP follows the publisher-subscriber paradigm, where the publisher is called talker and the subscribers, listeners. The real-time data communications are made through streams. A stream is a logical communication channel that carries traffic defined by a set of parameters, such as the period or frame size. For example, if one temperature sensor, the talker, wants to transmit its measurements with a period of 10 ms and a payload of 1 byte to other nodes, the listeners, the network has to check that there are enough resources and, if there are, it must create a stream with the specified period and payload.

As we have already said, there are three different SRP architectures. Nevertheless, as this work focuses on the distributed version of SRP, next we only explain the resource reservation mechanism of this version. Further details on the other architectures can be found in [7].

It is important to note that all the decisions regarding the reservation of resources are taken using local information only. Nevertheless, there is important information related to the reservations that must be propagated throughout the network; e.g. the amount of resources needed for a stream, whether a certain bridge has resources available or not, etc. This information is conveyed within special messages called talker and listener attributes. The arrows in Figure12.3 represent the direction in which the attributes are propagated throughout the network. Next we describe the process in detail.

Figure12.1 shows the time diagram of the resource reservation mechanism in a network with a line topology. This network consists of one talker (T), one listener (L) and two bridges (B1 and B2). As we can see in the figure, when a talker wants to transmit a set of frames with certain parameters, it must first create the stream to convey such frames. To create a stream the talker has

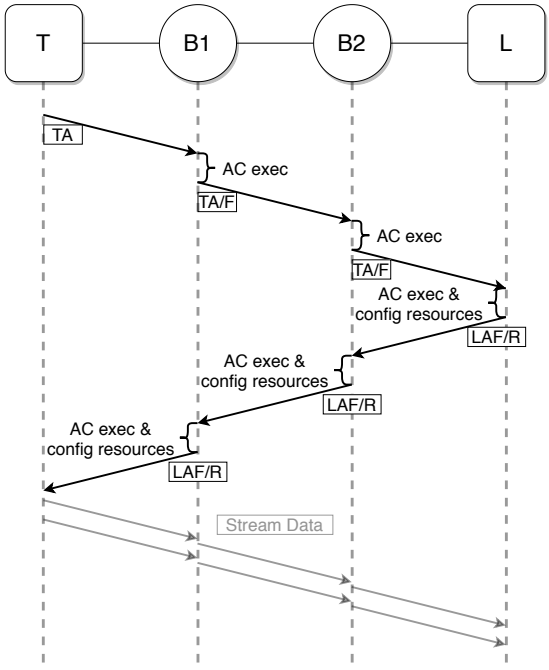


Figure 12.1: Time diagram of the resource reservation mechanism in a network with a line topology.

to declare its intention to communicate by transmitting in broadcast mode a special message called Talker Advertise (TA) message. This message conveys stream identification information, as well as the resources needed to convey the traffic. This information is then used in the rest of devices of the network to check whether there are enough resources for the stream so that it can be created. This evaluation of available resources is called Admission Control (AC). Note that SRP relies on other mechanisms that eliminate the loops in the network to prevent the TA message from circulating the network indefinitely.

The TA message transmitted by the talker is received by the bridge connected to it. When a bridge receives a TA message, each forwarding port checks if it has enough resources for the stream or not by executing the AC. In this protocol, a forwarding port is any port through which the TA message was not received, e.g., the port that connects B1 to B2 in Figure12.1. If the port has enough resources, the TA message is forwarded to the next device, i.e., the next bridge or node. On the other hand, if the port does not have enough resources, it sends a so called Talker Failed (TF) message instead. A TF message conveys the same information as the TA message plus the reason for the failure in the

reservation. Bridges that receive a TA message transmitted by another bridge through one of their ports behave as we have just described. In contrast, if the message received is a TF message, bridges transmit a TF message through all their forwarding ports, without carrying the AC.

Regarding nodes, we have to note that not all nodes are listeners for all streams. Therefore, if a node that does not want to become a listener of the stream receives a TA or TF message, it does not carry any further actions. In fact, it does not even inform the talker about its lack of interest in the stream. On the other hand, if a node receives a TA or TF message and is willing to listen to the stream there are three possible scenarios to consider: (i) the listener receives a TF message and cannot therefore become a receiver of the stream that is being created, so it sends a message called Listener Asking Failed (LAF) to the bridge; (ii) the listener receives a TA message but, while checking its resources it realizes that it does not have enough resources to receive the stream, so it sends an LAF message to the bridge; and, (iii) the listener receives a TA message and, while checking its resources it realizes that it has enough resources to receive the stream, so it sends a message called Listener Ready (LR) message to the bridge.

Each port of the bridges connected to a listener can receive an LR or LAF message. If a port receives an LAF message it does nothing else. If a port receives an LR message the port checks its resources again. If it does not have enough resources the port changes the LR received to an LAF; otherwise, if it has enough resources, the port reserves the resources (*config resources* in the figure). Whenever a bridge is connected to several bridges or nodes, it may have several listener responses to forward. In this case the bridge combines the responses into a single one and transmits it towards the talker. The result of combining the responses is the following: (i) if the bridge receives an LR in all the ports, it transmits to the talker an LR message; if the bridge receives an LAF in all the ports, it transmits to the talker another LAF message; and, if the bridge receives LR messages in some ports and LAF messages in other ports, it will transmits to the talker a new message called Listener Ready Failed (LRF) message. Whenever a bridge receives an LRF message it forwards an LRF message to the talker, regardless of the other listener attributes it receives. Note that in Figure12.1 listener attributes cannot be LRF because in a linear topology there is only one port receiving responses, therefore, bridges cannot receive LR messages through some ports and LAF messages through other ports.

Finally, the talker waits until it receives an LR or LRF message to start the data transmission. Once the stream has been created, the talker can delete it at any time by means of the unadvertise stream mechanism. The talker trans-

mits a message to eliminate the stream from all devices. This message is also transmitted in broadcast mode to ensure that all bridges and listeners receive the indication to eliminate the stream.

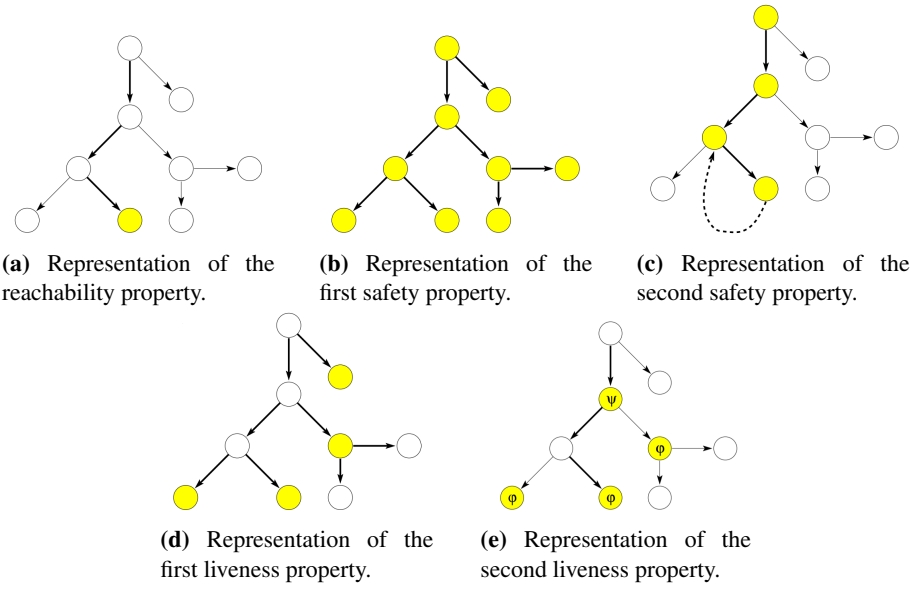
We call a reservation distribution to each possible combination of paths that reserve resources. We need to note that we consider to be good and bad reservation distributions. For instance, let us assume that we have a network with a star topology with one talker, two listeners and one bridge in the middle. One example of good reservation distribution is one where all the ports have reserved the required resources, whereas one example of bad reservation distribution is one where the listener ports have reserved the required resources but the talker port has not. In this last case there is a waste of resources in the links to the slaves because, as the port of the talker is not reserved, the slaves are not going to receive anything from that talker, regardless of their reservations.

12.4 SRP Uppaal Model

This section introduces the model developed in this work. The model is implemented using the UPPAAL model checker which is a tool for modeling real-time systems and formally verifying their properties [11]. In UPPAAL the systems are modeled by means of interconnected timed automata (finite-state machines extended with clocks that progress at the same pace). Each automaton is specified by a template that can be instantiated several times. At the same time, templates are constructed using locations, edges, local variables and local clocks, and can synchronize through different types of channels.

In addition, UPPAAL provides a formal query language that allows defining properties that the system should exhibit. These properties can be classified into 3 types and 5 sub-types represented in Figure 12.2. In this figure, circles represent states of the model whereas filled ones represent states with certain characteristic. The first property type, shown in Figure 12.2a, is the reachability property. It checks if it exists any state with a specific characteristic, e.g., a state in which the message arrived to the listener correctly. The second one is the safety property. It checks if a specific characteristic is in all the states (Figure 12.2b) or in all the states of a path of the state space (Figure 12.2c), e.g., it can check if, in all the states of the state space, messages are not lost or are not affected by errors. Finally, the third property type is the liveness property. It checks if a state with specific characteristics is eventually reached (Figure 12.2d) or if the state (φ) is eventually reached after another state (Ψ) with another characteristic was reached (Figure 12.2e).

Using the model and the queries as inputs, the tool performs an exhaustive check of the properties, i.e., it explores all the possible execution paths of the



(a) Representation of the reachability property.

(b) Representation of the first safety property.

(c) Representation of the second safety property.

(d) Representation of the first liveness property.

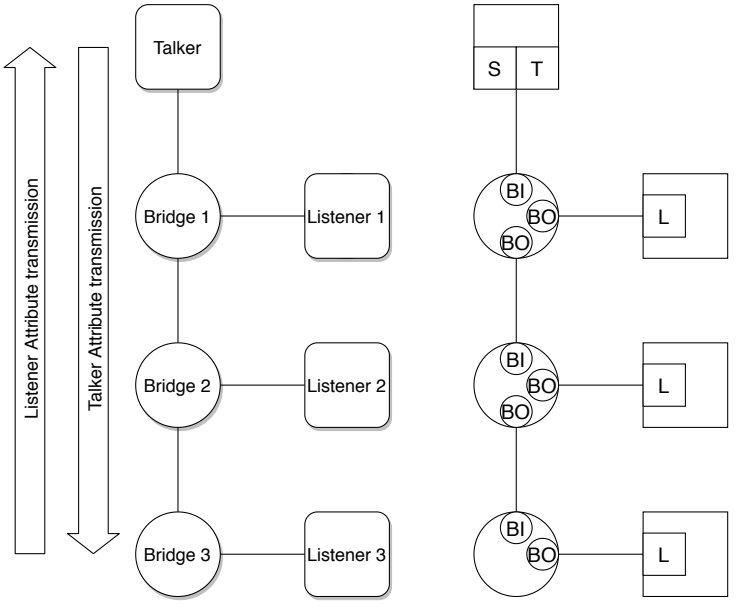
(e) Representation of the second liveness property.

Figure 12.2: Types of properties that can be evaluated in UPPAAL based on a figure from [11]. Each figure shows a representation of the state space of the model. The filled circles are the states that exhibit certain characteristic.

model to verify whether the properties hold. After this, UPPAAL informs the user about the result and, if a property does not hold, it shows an execution path in which the property is violated.

In this paper we abstract the description of the model. Section III of [14] describes our model in an exhaustive way. The complete model files and the files used for its verification can also be found in [14].

Figure12.3 (a) represents the network we modeled with UPPAAL while Figure12.3 (b) represents the resulting UPPAAL model. As can be seen, our SRP model is made of 5 different templates: Talker template, Stream template, Listener template, BridgeInput template and BridgeOutput template (represented as T, S, L, BI, BO respectively in Figure12.3(b)). These templates model the different relevant actions of the protocol carried out by the talkers, bridges and listeners. Specifically, as we can see in Figure12.3, our model has one instantiation of the Talker and Stream templates to model the actions carried out by one talker. It also has three instantiations of the Listener template to model the actions carried out by three listeners. And, finally, it has 3 instantiations of the BridgeInput template and five instantiations of the BridgeOutput template to model the actions carried out by three AVB bridges. Other network elements, such as links, are represented in the model by variables,



(a) Modeled network consisting of one talker, three listeners and three bridges.

(b) Abstraction of the network model made with UPPAAL.

Figure 12.3: Representation of the modeled network and its model by means of templates where T represents the Talker template, S the Stream template, BI the BridgeInput template, BO the BridgeOutput template and L the Listener template.

clocks and channels.

As we can see in Figure12.3(a), and as we have already said, our model is made up of one talker, three bridges and three listeners, each connected to one bridge. We decided to use three listeners for many reasons. The first reason is that, in many critical systems is usual to use active replication, using three replicas which perform majority vote on each result, in order to tolerate the failure of nodes. Moreover, three listeners are enough to have all relevant combinations of responses of the listeners. This is because, as listeners can only be or not interested in the stream or interested with or without resources, having more listeners would increase the times one of this options appears but would not produce a different scenario. On the other hand, we connected one listener to each bridge to have paths with different lengths and end-to-end delays, factors that increase the likelihood of encountering consistency issues. Finally, we used a line topology because SRP relies on other protocols that

eliminate the loops of the network, such as the Rapid Spanning Tree Protocol [1] or the Shortest Path Bridging Protocol [6].

Like any model of a system, our SRP model has a series of abstractions. First, we only model the transmission of one stream because allowing the model to transmit several streams would lead to the explosion of the state space without providing any benefit, on the contrary, it would make the model more difficult to analyze. We neither model the transmission of data frames because it is not part of SRP and it would increase the complexity of the model unnecessarily, as it would distort the model without giving greater precision to the analysis of the protocol. Finally, we did not take into account the presence of errors for several reasons. First, the property issues we detected appear in the absence of faults in the network. Secondly, there are some works like the one presented in [10] that allow tolerating faults in the channel by using proactive replication of frames. It is important to note that this level of abstraction is typical for network models and that we have validated our model to ensure that it is a faithful representation of SRP.

The model used in [13] abstracted the number of ports of the devices. Specifically, all devices had a single reception port and a single transmission port, even though AVB bridges can have an undetermined number of ports. This abstraction reduced the state space and the complexity of the Bridge template, and, in addition, avoided specifying a specific number of ports on the bridges which would decrease the generality of the model. However, this abstraction moved slightly away from reality, reduced the information present in the bridges and transferred certain decisions from the bridges' output ports to the input ports of the devices connected to them.

In this work we present a more detailed, yet analyzable and general model. Specifically, our model divides the Bridge template into two, one for the reception port of the talker attributes and transmission of the listener attributes and another for the reception of the listener attributes and transmission of the talker attributes. These templates can be instantiated as many times as necessary for each bridge, so the generality of the model is maintained. In addition, the model conforms more to reality and allows to keep decisions and information on the bridge. Nonetheless, this new model has an increased state space compared to the one in [13], which increases the time required for the analysis.

12.5 Evaluation of the Termination of SRP

As we said in Section 12.1, termination is a basic property of the CDESs. Thus, all TSN protocols should provide it in order to support this kind of systems,

even more if we talk about SRP which is responsible to accept the streams and reserve the resources, a key piece to ensure a good behavior of the system.

This analysis does not aim at demonstrating that the distributed version of SRP does not present termination, since this is relatively obvious once we know the protocol. Instead, we use it to analyze in depth the cases of non-termination and their consequences. In addition, this analysis is helpful to find a solution to the problem and to have a reference to evaluate the proposed solution.

In this work we differentiate two levels of termination: termination for the application and for the infrastructure. The first one affects the nodes and, therefore, the application. The lack of termination at the application level can cause malfunction of some critical applications. This is due to the fact that many of those applications require to know the result of the reservation to make important decisions.

The infrastructure level refers to the bridges of the network. Even if in an ideal system these devices do not require termination, it is important to provide it to prevent unforeseen and undesirable effects in future reservations. For example, if a bridge receives many requests without resolution, it would be possible to cause an overflow of the buffer that could prevent the bridge from accepting new reservation requests or force it to eliminate some already accepted ones.

We next present the problems detected but it is important to note that the issues are mainly due to the fact that in SRP listeners do not inform the bridges nor the talkers when they are not interested in binding to a stream. Section IV of [14] shows and explains in more detail the queries used.

12.5.1 Termination at the Application Level

Using the UPPAAL model, we find a series of scenarios where the talker does not receive any response from the listeners and, thus, it waits indefinitely. This can happen even in the absence of faults, when there are no listeners interested in the stream. As we have said before, many critical applications require to know the result of the reservations to make important decisions. Thus, the lack of termination can cause a malfunction of those applications, such as blocking the decision process or leading to incorrect decisions due to the lack of knowledge.

To check the termination for the application level and determine the causes of the issues detected we used the query (Q1), which corresponds to query 1 (safety property) in [14]. This query checks if there is a path of states in the system in which the talker never receives any listener response. The query

is satisfied which means that there are scenarios in which a talker does not receive any listener response leading to a termination issue. Then we checked if it is possible that this happens if at least one listener is interested in the stream. To do that we used another query (Q2), which corresponds to query 2 (liveness property) in [14]. This query checks if, at the end of the listeners actions, the response of at least one listener, i.e., there is at least one listener interested in the stream, implies the reception of responses by the talker. This query is satisfied, which means that if at least one listener is interested in the stream, the talker receives at least one response. Finally, we checked that the non-reception of responses by the talker was due to the non-transmission of responses by the listeners. This was checked with another query (Q3), which corresponds to query 3 (liveness property) in [14]. This query checks if, at the end of the listeners actions, if no listener has responded, the talker receives no response.

The use of the previous queries allowed us to determine that the only case where termination is not achieved in the application level is when no listener is interested in the stream that the talker is announcing.

12.5.2 Termination at the Infrastructure Level

A bridge that forwards the request of a talker waits for the responses of the listeners indefinitely. Also, bridges register talkers' attributes in all their ports, and they do so for all the talkers willing to transmit. Similar to what happens for termination at the application level, we find some scenarios where some bridges do not receive any response from the listeners, even in the absence of faults and even if the first level of termination is actually achieved by the protocol. Thus, bridges can wait indefinitely, e.g., if there are no listeners interested in the stream connected directly or indirectly to the bridge. This can cause an unnecessary use of memory in bridges and can later prevent the creation of streams with listeners willing to bind due to a lack of memory.

To check the termination at the infrastructure level, and determine the causes of the issues detected, we used three different queries for each port. These queries are similar to the ones used in the verification of the termination at the application level, but have the particularity that must be checked for each port of the bridge. The first of these three queries (Q4), which corresponds to queries 4, 7, 10, 13 and 16 (safety property) in [14], checks if there is any path of states in the system in which the port of the bridge does not receive any listener response. As the query is satisfied, it is possible that the port of a bridge does not receive any listener response, leading to a termination issue in the bridges.

Then, we used another query (Q5), which corresponds to queries 5, 8, 11, 14 and 17 (liveness property) in [14], to verify if this is possible even if at least one listener connected directly or indirectly to the bridge is interested in the stream. This query checks if the response of at least one listener, i.e., if at least one listener is interested in the stream, leads to the reception of responses by the bridge port. As this query is satisfied, if at least one listener is interested in the stream, the port of the bridge receives at least one response.

Finally, we checked that the non-reception of responses by the port of a bridge was due to the non-transmission of responses by the listeners connected directly or indirectly to the port of the bridge. This was checked with the third query (Q6), which corresponds to queries 6, 9, 12, 15 and 18 (liveness property) in [14]. This query checks if when no listeners have responded at the end of the listeners actions, so there are no listeners interested in the stream, the port of the bridge receives no response.

The use of these queries allowed us to conclude that the only case where there is no termination at the infrastructure level is the one where no listeners connected directly or indirectly to a bridge are interested in the stream.

12.6 Evaluation of the Consistency of SRP

As it was introduced in Section 12.1, consistency is an important property in CDESs. For instance, SRP should do the reservation of resources in a consistent manner if we intend to have several replicated nodes. The nodes should have the same inputs and outputs, which cannot be guaranteed without a consistent reservation of resources. This is just a simple example that illustrates the importance of consistency, even though consistency can be required in any distributed system.

As in Section 12.5, this analysis was not performed to demonstrate that the distributed version of SRP does not present consistency, which is relatively obvious once you know the protocol, but to analyze in depth the scenarios that can cause inconsistencies and their consequences. In addition, again, this analysis is helpful to find a solution to the problem and to have a reference to evaluate the proposed solution.

As in the previous section, we differentiate two levels of consistency: consistency for the application level and for the infrastructure level. Again, the first one affects the nodes and, therefore, the application. The lack of consistency at the application level can cause malfunction of some critical applications. Some of the applications targeted by TSN require the different nodes to carry out coordinated actions because, e.g., they may rely on active replication of the nodes. In these applications, consistency in the communications is key to

guarantee the correct operation of the overall system. The first step towards achieving consistent communications is to reserve the network resources consistently. Thus, at this level, SRP should guarantee that enough listeners have resources reserved for the communication before starting to transmit.

As before, the infrastructure level refers to the bridges of the network. As we will see later, inconsistencies when reserving resources in bridges can cause the waste of resources. This, in the long term, causes that streams, for which there would be sufficient resources, cannot be declared due to the resources reserved and wasted in some bridges.

As in the evaluation of the termination, despite the importance of consistency, we found some issues in both levels even in the absence of faults. We next present the problems detected but it is important to note that the issues are mainly due to the fact that information related to the reservations is propagated in a single direction. That is, the talker attribute transmitted by a talker is forwarded always towards the listeners; while, when listeners and bridges reply to a stream declaration, the information is only forwarded towards the talker. Thus, not all the devices involved in the reservation of a stream receive the same information. We next describe the consistency issues detected and their effects. Section V of [14] shows and explains in more detail the queries used.

12.6.1 Consistency at the Application Level

In SRP, resources can be reserved for a subset of listeners, even when there are listeners willing to communicate that do not have resources to do it. In this case, the talker only communicates to a subset of listeners, generating an unnoticed inconsistency in the exchange of data. This means that actually starting a stream (with some listeners) has priority over doing it consistently (with either all or none of them). In addition, talkers cannot know which listener has enough resources and which one does not. A talker only knows if all interested listeners have enough resources when it receives LR messages; if all interested listeners have not enough resources when it receives LAF messages; if no listener is interested when it does not receive any answer; or, if at least one interested listener has enough resources when it receives LRF messages. This limited information does not allow the talker to take intelligent decisions. Furthermore, we have to take into account that this information can change during the execution of the SRP mechanism e.g. it is possible for a talker to receive an LR message and then receive an LRF message. Something similar can happen in listeners. They may be interested in the stream and have sufficient resources, but they do not receive anything because during the transmission of

the response, the route to the talker did not have enough resources.

Furthermore, even when all listeners willing to bind have enough resources to do so, there are scenarios where consistency for the application is not guaranteed all the time. This can happen for two reasons, first the paths between a talker and different listeners may differ in length and end-to-end delay and, second, the talker starts transmitting as soon as it receives the response of one listener ready to receive. Therefore, some listeners willing to bind to the stream, with enough resources throughout the whole path towards the talker, may miss the first frames transmitted by the talker. This can cause, for example, two replicas to be in two different states so that, although from that moment they receive the same data, they will not provide the same result.

To check the consistency for the application level, and to determine the causes of the issues detected, we used query (Q7), which corresponds to queries 19, 20 and 21 (reachability property) in [14], to check if there is at least one state in which the talker is already transmitting, a listener is interested in the stream and, from his point of view, has sufficient resources but the route from the talker to the listener has not reserved the necessary resources for that stream. This test shows that the talker can start transmitting even when there are interested listeners that will not be able to receive the stream. Moreover, it also shows that there are listeners that believe they are going to receive the stream but never will.

Another query (Q8), which corresponds to query 22 (liveness property) in [14], is used to verify that, even when all interested listeners can bind to the stream, some of them may miss the first messages because the talker starts transmitting before finishing the resource reservation. Specifically, it checks if a talker transmitting, a listener waiting for the stream and the route not yet reserved may lead the system to a state in which the route will never be reserved. As the query is not satisfied we proved the inconsistency in the data received at the beginning of the stream.

12.6.2 Consistency at the Infrastructure Level

In this work we also find out that bridges can make inconsistent decisions regarding the reservation of resources of a stream. Specifically, in SRP it is possible that some bridges reserve resources for a stream but other bridges in the same route to the listener do not. This implies a waste of resources in the bridges that reserved the resources because the listener for which they reserved the resources is not going to receive the stream because of the bridges in the same route that did not reserve the resources. This may not be problematic at first, but, with an utilization close to 100%, this may cause streams, for which

there would be sufficient resources, cannot be declared due to the resources wasted in these bridges.

To check the consistency for the infrastructure level, and to determine the causes of the issues detected, we used query (Q9), which corresponds to queries 23 and 24 (reachability property) in [14], to check if there is at least one state, after the mechanism has been executed, in which the stream is being transmitted while the link that supplies one or more bridges is not reserved but the links of the bridges are. This reservation distribution implies a waste of resources in all the links reserved by the bridges affected because, as the link that supplies them is not reserved, they are not going to receive data messages from this stream. Finally, another query (Q10), which corresponds to query 25 (liveness property) in [14], checks if the transmission of the stream always leads to one of all correct distributions of resource reservations. As it is not satisfied, we can determine that incorrect distributions of resource reservations (with waste of resources) may happen using SRP.

12.7 CSRP Description

We designed a series of solutions to deal with the drawbacks described in Sections 12.5 and 12.6. The main objective is to provide network devices with a consistent view of the reservation of resources so that they can make rather complex decisions within a bounded time.

A trivial solution could consist in modifying the current SRP in the following way:

1. The talker multicasts the talker attribute instead of broadcasting it. With this change, the network avoids sending the talker attribute to non-interested listeners, eliminating the termination issue of the application and infrastructure level. The problem is that it makes the network less open and adaptive, understanding open as a network where nodes can join or leave a stream dynamically and adaptive as a network where stream requirements change during run-time.

2. A bridge that has decided that it has enough resources when retransmitting the talker attribute cannot change this decision when it receives the listeners' responses. This would allow bidirectional propagation of decisions without adding another round of transmissions solving the consistency issue at the infrastructure level. The problem is that it can hinder the creation of streams that attempt to be declared simultaneously and does not prevent the waste of resources in unnecessary reservations.

3. The listeners multicast the listener responses, instead of unicasting them, conveying their ID in the response. In this way the listeners can inform the talker, the bridges and other listeners of whether they can receive or not, solving the consistency issue at the application level.The problem is that listeners can flood the network with control messages because each of them would transmit its status in multicast mode.

4. The talker, listener and bridges must make decisions deterministically based on the information received which, in the absence of faults, will be consistent.

Despite solving the problems detected, this solution greatly modifies the mechanism and has several limitations. For these reasons we decided to develop the following solution which is a compendium of the different solutions proposed in [13].

The most relevant differences between this new protocol and previous proposals are:

1. The listener responses convey the ID of the listeners that sent them. This gives the talkers and bridges the necessary information to know which listeners can receive and which not.

2. Bridges' and listeners' decisions can change depending on the decision taken by the talker.

3. The talker, after a bounded time limited by a timer in it, will decide which listener can receive the stream and which cannot based on the information received. After this, the talker will send the result of his decision to all the devices so that they have a consistent view of the status of the reservations and carry out the necessary actions.

Even if this solution requires some changes in SRP, it can be implemented in a way that non-modified devices can still work normally. For example, in the implementation we are proposing next, listeners do not need to implement the solution although this would imply that these listeners would not have a consistent view of the network. Furthermore, it provides the talker the necessary information to make a centralized, rather complex and bounded in time decision which, in addition, will be sent to the rest of the network to maintain the consistent view of the reservation of resources. Moreover, all this would be achieved by adding only one additional broadcast transmission, the one by the talker with the final decision, avoiding the multiple multicast transmissions of the listeners added in the previously proposed trivial solution.

The name of this new protocol is CSRP and its resource reservation mechanism is as follows:

The transmission of talker and listener attributes proceeds as in SRP, as it is described in Section 12.3. The first modification of the protocol is found in the transmission of listeners attributes by the bridges. Bridges receive the listener attributes and combine them to send them to the talker. In order to accomplish this, bridges analyze the responses received by each port and then generate the new response that they transmit towards the talker. Whenever a bridge receives an LR message through a port, it checks whether the port has enough resources. If there are enough resources, the LR remains unchanged and the port reserves the necessary resources provisionally, instead of definitely like in SRP; otherwise, the LR becomes an LAF message. On the other hand, if the bridge receives an LAF message the value is left unchanged and the port does not reserve the resources. In case of concurrent requests, and this is another change with respect to SRP, the provisional reservation is made for the first LR or LRF message received, while the rest are transferred to a First-In, First-Out (FIFO) list. The items in this list are only deleted when their reservation processes are completed or when the reservation of resources is confirmed.

After processing the listener attributes, each bridge must join them to forward an updated one to the talker. This process is the same as in SRP and is described in Section 12.3. It is important to note that bridges do not wait for the reception of all the listener attributes, but they are continuously joining and retransmitting them as they receive new answers. In this way a bridge can transmit an LR or LAF message and then transmit an LRF message, just like in SRP. Nevertheless, in CSRP bridges must specify in the listener attribute which listeners can receive and which listeners cannot. To do so, CSRP relies on two lists, one for successful reservations and one for unsuccessful ones. Specifically, edge bridges introduce the identifier of the node that sends the LR or LAF message in the corresponding list and sends them embedded in the response to the talker. Whenever a bridge receives a response from another bridge, it checks the lists and updates them accordingly when joining the responses.

The talker waits for the answers for a bounded period of time. This time will depend on the application, topology and size of the network. Basically, it must ensure that the response from the farthest slave has enough time to reach the talker. This is implemented by means of a local timer in the talker which is activated at the beginning of the transmission of the TA and expires after the predefined time. After that time, the talker uses the lists with the nodes identifiers to know which listeners can receive and which listeners cannot and it decides whether to transmit the stream to all the listeners that can receive, to

a subgroup or to none of them. This decision is communicated by transmitting in broadcast mode a message called Final Decision (FD), which contains a list of listeners that will receive the stream and listeners that will not receive the stream.

When a bridge receives the FD message it knows which listeners must receive the stream and which must not. In this way, bridges can lock the resources or eliminate unnecessary reservations. Listeners, on the other hand, can know whether they are subscribed to the stream or not and do not wait indefinitely for the data transmission.

Once the FD message has been transmitted and the resource reservation mechanism has finished, the talker starts transmitting the data stream. Finally, as in standard SRP, once the stream has been created, the talker can delete it at any time by means of the unadvertised stream mechanism.

12.8 CSRP Uppaal Model

The UPPAAL model of CSRP has the same topology, same templates, same instantiations of the templates and same abstractions as the model of the standardized SRP explained in Section 12.4. To formally verify the correction of the improved mechanism (CSRP's resource reservation mechanism), we modified as little as possible the model shown above to include the changes proposed in our solution. Again, the templates and the modifications applied to them can be found in Section VII of [14]. The complete model files and the files used for its verification can also be found in [14]. We next explain in an abstract way the main changes done to the SRP model.

In the Talker template we basically eliminated the instantaneous transmission of data that occurred as soon as the speaker received an LR or LRF message. On the other hand, we added a timer to define the waiting time for listener responses and implement the transmission of the FD message.

In the bridge templates we implemented the reception and forwarding of the FD message and the mechanisms to change the resource reservations based on it.

Finally, in the Listener template we implemented the reception of the FD message and the mechanism so that listeners know if they can receive or not. It is important to remember that these modifications in listeners are not essential. However, not implementing them would imply that listeners remain unsure of whether they will receive or not until they receive any data message of the stream.

12.9 Evaluation of the Termination of CSRP

We next describe the verification of CSRP from the termination point of view. Again, we address the issues at the application and infrastructure level. To do that, we used the same queries that proved the non-termination in SRP plus some additional queries. These are explained in more detail in Section VIII of [14].

Just like in SRP, if in CSRP no nodes want to bind to a stream, the talker and bridges do not receive any listener responses. Thus, we provide termination with the timer in the talker and the FD message in the bridges, as now both, talkers and bridges, know when to stop waiting for listener responses.

12.9.1 Termination at the Application Level

In CSRP it is still possible that the talker does not receive any response from the listeners (see Sub-section 12.5.1). However, using the timer, the talker always stops waiting for an answer, makes a decision based on the information it has received and informs about it by means of the FD message to the rest of the network. We used the UPPAAL model of CSRP to verify the behavior of the protocol. Specifically, we check that all the nodes finish the resource reservation process within a bounded time determined by the timer in the talker and the distance between the talker and the listeners.

To check the termination at the application level of CSRP we used the same three queries that we used to evaluate SRP (Q1, Q2 and Q3), which correspond to queries 26, 27 and 28 in [14]. The results of evaluating these queries allow us to determine that in CSRP it is still possible that the talker never receives any response from the listeners. However, we used another query (Q11), which corresponds to query 29 (liveness property) in [14], to evaluate whether the talker stops waiting for a response after the timer has expired, makes a decision based on the received information and informs about it by means of the FD message to the other devices of the network. This query proved that CSRP provides termination at the application level.

12.9.2 Termination at the Infrastructure Level

With this evaluation we see how bridges' ports may not receive any listener response. However, thanks to the FD message sent by the talker, bridges always stop waiting for an answer and change their reserved resources based on the talker decision. We used the UPPAAL model of SRP to verify that all bridges finish the resource reservation process within a bounded time.

Using the same queries as the ones used to evaluate SRP (Q4, Q5 and Q6), which correspond now to queries 30 to 36, 38 to 40, 42 to 44 and 46 to 48 in [14], we found that bridges' ports may not receive any listener response. However, another query (Q12), which corresponds to queries 33, 37, 41, 45 and 49 (liveness property) in [14], demonstrated that, thanks to the FD message, the bridges always stop waiting for an answer and change their reserved resources based on the talker decision. Thus, CSRP provides termination at the infrastructure level.

12.10 Evaluation of the Consistency of CSRP

In this section we describe the verification of CSRP from the consistency point of view, at the application and infrastructure level. To do so, we use the same queries that proved the inconsistency in SRP plus some additional queries. These are explained in detail in Section IX of [14]. Nevertheless, this solution solves all the detected consistency issues. We achieved this by centralizing the decisions in the talker and ensuring the homogeneous propagation of information related to the reservation of resources.

12.10.1 Consistency at the Application Level

First, note that this solution does not aim at providing resources for all the listeners that want to bind. Instead, it aims at ensuring that all listeners know what is the status of the reservation regardless of whether they can receive or not. This was not guaranteed in the standard SRP but it is achieved in CSRP thanks to the FD message. We verify the consistent view of the network. Specifically we prove that when CSRP finishes the reservation process, all devices know which nodes are subscribed to the stream and which are not, including the nodes.

By using query Q7, which corresponds now to queries 50, 53 and 56 in [14], also used to evaluate SRP, we found that in CSRP there are states where a listener wants to bind to the stream and it thinks it can but the resources have not been reserved. However, another query (Q13), which corresponds to queries 51, 52, 54, 55, 57 and 58 (reachability and safety properties) in [14], demonstrated that, thanks to the FD message, now listeners know when they can and when they cannot receive the stream.

Finally, we used two queries (Q14 and Q15), which correspond to queries 59 and 60 (safety property) in [14], to verify the consistent view of the network. These queries verify that always at the end of the reservation process

all the lists with the successful and unsuccessful reservations explained in Section 12.7 are consistent. Therefore, we can conclude that CSRP provides consistency at the application level.

12.10.2 Consistency at the Infrastructure Level

Finally, at the infrastructure level, we verify that CSRP avoids wasting resources with unnecessary reservations thanks to the FD message that informs the bridges about which listeners have been able to bind to the stream and which listeners have not, so that the bridges can free the resources they reserved for the listeners that cannot receive. We carry out this verification using the CSRP UPPAAL model.

To check the consistency at the infrastructure level of CSRP we used queries Q9 and Q10, which correspond now to queries 61, 62 and 63 in [14], also used in the evaluation of SRP. These queries prove that, at the infrastructure level, not only we avoid wasting resources with unnecessary reservations but also only the appropriate reservation distributions are generated. Moreover, we can conclude that CSRP provides consistency at the infrastructure level.

12.11 Conclusions

The IEEE AVB TG defined a series of standards to provide Ethernet with soft-real time capabilities. Later on, the TG broadened its scope and was renamed to TSN. The new TG aims at providing hard real-time guarantees, flexibility of the network configuration and fault tolerance mechanisms to Ethernet. Therefore, the work carried out by the TSN TG is intended to enable the use of standard Ethernet as the network technology for critical distributed embedded systems. One of the most important projects of the TGs is SRP, as it allows to reserve resources to provide timing guarantees and prevent frame losses.

In this work we present a model of the AVB's distributed version of SRP in UPPAAL. Thanks to it, we identify multiple scenarios in which SRP does not exhibit termination nor consistency and we determine the possible adverse effects that their lack can cause. Moreover, we have proved that these properties can be violated even in the absence of faults. Some of the problems detected and their effects are: (i) talkers and bridges in most of the cases do not know whether the resource reservation process is over, (ii) talkers never know which listeners have bound to the stream and which ones have not, (iii) subscribed listeners can miss the first data messages of the stream and (iv) some bridges can waste resources because they don't know if the bridges they depend on

(bridges located on the route between the talker and them) have been able to reserve resources for them.

Thus, in this work we propose an enhanced version of SRP that enforces termination and consistency, thereby eliminating all the issues we have identified for SRP. This solution is called CSRP and not only it solves the aforementioned problems, but it also provides the nodes and bridges with enough information to make rather complex decisions about the reservation of resources within a bounded time. For instance, now a talker can advertise a stream and know which listeners can receive and which ones cannot, after a certain time determined by an internal clock. Based on this information, the talker can decide whether all, a subset or none of the nodes that manifest their interest in the stream can subscribe.

Finally, we have developed a UPPAAL model of CSRP and we have used it to verify that CSRP provides termination and consistency to the reservation of resources in the absence of faults. Further work will be done to extend the study of the behavior of CSRP in other scenarios and even using an experimental implementation to evaluate new aspects, such as its performance in real networks.

Bibliography

[1] IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges. *IEEE Std 802.1D-2004 (Revision of IEEE Std 802.1D-1998)*, pages 1–281, June 2004.

[2] IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams. *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pages C1–72, Jan 2009.

[3] IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP). *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*, Sept 2010.

[4] IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks. *IEEE Std 802.1AS-2011*, pages 1–292, March 2011.

[5] IEEE Standard for Local and Metropolitan Area Networks–Audio Video Bridging (AVB) Systems. *IEEE Std 802.1BA-2011*, pages 1–45, Sept 2011.

[6] IEEE Standard for Local and Metropolitan Area Networks–Media Access Control (MAC) Bridges and Virtual Bridges. *IEEE Std 802.1Q, 2012 Edition, (Incorporating IEEE Std 802.1Q-2011, IEEE Std 802.1Qbe-2011, IEEE Std 802.1Qbc-2011,IEEE Std 802.1Qbb-2011, IEEE Std 802.1Qaz-2011, IEEE Std 802.1Qbf-2011,IEEE Std 802.1Qbg-2012, IEEE Std 802.1aq-2012, IEEE Std 802.1Q-2012*, pages 1–1782, Dec 2012.

[7] IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks – Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements. *IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018)*, pages 1–208, Oct 2018.

[8] YANG Modules. `https://1.ieee802.org/yang-modules/`, 2018.

[9] IEEE Audio and Video Bridging Task Group. `https://1.ieee802.org/tsn/`, April 2019.

[10] I. Alvarez, J. Proenza, and M. Barranco. Towards a Time Redundancy Mechanism for Critical Frames in Time-Sensitive Networking. In *Proceedings of the IEEE 22nd International Conference on Emerging Technologies and Factory Automation (ETFA 2017)*, January 2018.

[11] Gerd Behrmann, Alexandre David, and Kim G. Larsen. *A Tutorial on Uppaal*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[12] M Björklund, J Schönwälder, P Shafer, K Watsen, and R Wilton. NETCONF Extensions to Support the Network Management Datastore Architecture. Technical report, RFC 8526, ISSN: 2070-1721, `https://tools.ietf.org/html/rfc8526`, 2019.

[13] D. Bujosa, I. Alvarez, D. Čavka, and J. Proenza. Analysing Termination and Consistency in the AVB's Stream Reservation Protocol. In *Proceedings of the IEEE 24th International Conference on Emerging Technologies and Factory Automation (ETFA 2019)*, October 2019.

[14] Daniel Bujosa, Inés Álvarez, and Julián Proenza. Description of the UPPAAL Models for SRP and CSRP and Verification of their Termination and Consistency Properties. Technical report, arXiv:2007.15712 [cs.NI], `https://arxiv.org/abs/2007.15712`, 2020.

[15] A. Gothard, R. Kreifeldt, and C. Turner. AVB for Automotive Use White Paper. Technical report, AVnu Alliance, Oct 2014.

[16] M. D. Johas Teener, A. N. Fredette, C. Boiger, P. Klein, C. Gunther, D. Olsen, and K. Stanton. Heterogeneous Networks for Audio and Video: Using IEEE 802.1 Audio Video Bridging. *Proceedings of the IEEE*, 101(11):2339–2354, Nov 2013.

[17] O. Kleineberg, P. Fröhlich, and D. Heffernan. Fault-Tolerant Ethernet Networks with Audio and Video Bridging. In *ETFA2011*, pages 1–8, Sept 2011.

[18] H. Lim, D. Herrscher, and F. Chaari. Performance Comparison of IEEE 802.1Q and IEEE 802.1 AVB in an Ethernet-Based In-Vehicle Network. In *2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT)*, volume 1, April 2012.

[19] H. Lim, L. Völker, and D. Herrscher. Challenges in a Future IP/Ethernet-based in-car Network for Real-Time Applications. In *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 7–12, June 2011.

[20] L. Lo Bello. Novel Trends in Automotive Networks: A Perspective on Ethernet and the IEEE Audio Video Bridging. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–8, Sep. 2014.

[21] P. Meyer, T. Steinbach, F. Korf, and T. C. Schmidt. Extending IEEE 802.1 AVB with Time-Triggered Scheduling: A Simulation Study of the Coexistence of Synchronous and Asynchronous Traffic. In *2013 IEEE Vehicular Networking Conference*, pages 47–54, Dec 2013.

[22] Jörn Migge, Josetxo Villanueva, Nicolas Navet, and Marc Boyer. Insights on the Performance and Configuration of AVB and TSN in Automotive Ethernet Networks. In *9th European Congress on Embedded Real Time Software and Systems (ERTS 2018)*, Toulouse, France, January 2018.

[23] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury. Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research. *IEEE Communications Surveys Tutorials*, 21(1):88–145, Firstquarter 2019.

[24] D. Park, J. Lee, C. Park, and S. Park. New Automatic De-Registration Method Utilizing a Timer in the IEEE802.1 TSN. In *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, pages 47–51, Oct 2016.

[25] R. Queck. Analysis of Ethernet AVB for Automotive Networks using Network Calculus. In *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*, pages 61–67, July 2012.

[26] R. Salazar, T. Godfrey, L. Winkel, N. Finn, C. Powell, B. Rolfe, and M. Seewald. Utility Applications of Time Sensitive Networking White Paper (D3). Technical report, IEEE, Sep 2018.

[27] S. Samii and H. Zinner. Level 5 by Layer 2: Time-Sensitive Networking for Autonomous Vehicles. *IEEE Communications Standards Magazine*, 2(2):62–68, JUNE 2018.

[28] T. Steinbach, F. Korf, and T. C. Schmidt. Real-time Ethernet for Automotive Applications: A Solution for Future In-Car Networks. In *2011 IEEE International Conference on Consumer Electronics -Berlin (ICCE-Berlin)*, pages 216–220, Sep. 2011.

[29] S. Thangamuthu, N. Concer, P. J. L. Cuijpers, and J. J. Lukkien. Analysis of Ethernet-switch Traffic Shapers for In-Vehicle Networking Applications. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 55–60, March 2015.

[30] M. Wollschlaeger, T. Sauter, and J. Jasperneite. The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0. *IEEE Industrial Electronics Magazine*, 11(1):17–27, March 2017.