Mälardalen University
School of Innovation Design and Engineering
Västerås, Sweden

Thesis for the Degree of Master of Science in Computer Science with Specialization in Embedded Systems - 15.0 credits

TIME SENSITIVE NETWORK (TSN) CONFIGURATIONS ON NETWORK PERFORMANCE IN REAL-TIME COMMUNICATION

Dalila Alibegović
dac21001@student.mdh.se
Lejla Smajlović
lsc21003@student.mdh.se

Examiner: Saad Mubeen
Mälardalen University, Västerås, Sweden

Supervisor: Bahar Houtan
Mälardalen University, Västerås, Sweden

Company supervisor: Sara Andersson,
Volvo CE, Eskilstuna, Sweden

June 10, 2022

Abstract

The automotive industry is dealing with the challenge of adapting the vehicular distributed embedded systems' technology to the expectations of modern cars. Today, more and more complex embedded software functions need to be implemented to provide the modern applications of a vehicle. Modern cars today are capable of running multiple critical functions such as airbag control or anti-lock braking system along with entertainment applications with a lower level of criticality. Whereas the future of the vehicles envisions partly-to fully-autonomous vehicles. As a result, the industry is constantly seeking for technology which drives the fulfillment of the future of cars by means of connectivity, complexity and scalability. One of the challenges ahead is the growth in the size and bandwidth of the communicated data in the vehicles due to the increase in the number of sensing and actuation components and embedded functions, which require high-bandwidth and low-latency in-vehicle scalable communication. The Time-Sensitive Networking (TSN) set of standards proposed by the IEEE 802.1 TSN task group provides a promising communication technology for many application domains, including the vehicular systems. Though in practice, TSN still lacks proper software and hardware engineering frameworks that facilitate the design, development, and implementation of TSN-based systems. In this thesis, we develop a TSN-based vehicular distributed embedded system use case. We compare the network's performance under TSN configurations with the case of a conventional standard Ethernet network setup. Various complexities were met during the realization of this thesis and designing such a system. Several proposed solutions represent a possible basis for future work and research.

Keywords: TSN, Ethernet, Automotive Systems, Response Time, Jitter

Table of Contents

List of Figures

List of Tables

Acronyms

Term	Meaning
ECU	Electronic Control Unit
TSN	Time-Sensitive Networking
ST	Scheduled Traffic
GUI	Graphical User Interface
CPU	Central Processing Unit
CBS	Credit-Based Shaper
TAS	Time-Aware Shaper
CAN	Controller Area Network
MTBF	Mean Time Between Failures
MTTR	Mean Time to Repair
HMI	Human Machine Interfaces
QoS	Quality of Service
LAN	Local Area Network
OSI	Open Systems Interconnection Model
ISO	International Standards Organization
MAC	Media Access Control
LLC	Logical Link Control
CSMA/CD	Carrier sense multiple access/collision detection
AVB	Audio-Video Bridging
IEEE	The Institute of Electrical and Electronics Engineers
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
BE	Best Effort Traffic
SRP	Stream Reservation Protocol
ATS	Asynchronous Traffic Shaping
CSR	Configurable Stream Reservation
SRP	Stream Reservation Protocol
GCL	Gate Control List
FIFO	First In First Out
SPR	Stream Reservation Protocol
SP	Strict Priority
TT	Time-Triggered traffic
AUTOSAR	Automotive Open System Architecture
OPC UA	Open Platform Communications United Architecture
SLOC	Source Lines of Code
SECU	Synonim for mutual hardware platform (Switch and ECU)
USB	Universal Serial Bus
HD	High-Definition
LED	Light-Emitting Diode
PC	Perosnal Computer
RGB	color-spaces (R- red, G - green, B - blue)
ICMP	Internet Control Message Protocol
IPv4	Internet Protocol version 4
RTOS	Real-Time Operating System

1. Introduction

High-bandwidth and low-latency in-vehicle communication has become a need in modern automotive systems due to the functionality advancements, e.g., in autonomous driving and greater user interactivity. Standard Ethernet cannot guarantee the timing requirements of the communication in case of high network utilization because it lacks mechanisms for bandwidth reservation. As a solution to those problems, TSN is proposed. High bandwidth, compatibility, and scalability are the reasons why Time-Sensitive Networking (TSN) is a matter of interest mainly in the automotive industry [1]. Since TSN is primarily new in the industry, there are just a few hardware and software implementations. In practice, adopting the TSN using already proposed solutions raises new challenges (Check 3.). Realization of a simple framework is essential for wide adaption of TSN in the industry [2].

A distributed embedded system is an embedded system in which the system's functionality is distributed over multiple nodes (end stations) connected via a bus or a network [3]. Without extensive testing, end-to-end response time and latency analysis can verify timing constraints given for vehicular distributed embedded systems [4][5]. Examples of those constraints are age and reaction constraints. Age constraint indicates "how long before each response a corresponding stimulus must have occurred" [6]. Reaction constraint indicates "how long after the occurrence of a stimulus, a corresponding response must occur" [6]. One of the biggest challenges for developers is to create a system capable of meeting deadlines under any conditions. To succeed, the developers have to do accurate calculations and build the appropriate task scheduling system. If it is possible to prove or demonstrate, during the design phase, that a system will meet all specified timing requirements during the execution, then the system is predictable [1][7]. Since the number of TSN network nodes in distributed automotive systems rises, and not all of them are using the same TSN classes nor standards, doing timing analysis becomes a challenging job. A possible solution to this problem is an implementation of the end-to-end data-propagation latency analysis [2]. Since all nodes in large-scale networks can use different TSN standards depending on the required mechanisms for a specific task, the configuration task represents a real challenge. Even configuring just one switch with two classes represents a complex assignment since TSN is primarily new in the industry, and there are not so many hardware and software implementations that can be used. For these reasons, the possibility of overcoming the mentioned challenges motivates researchers to continue investigating the complexities of implementing TSN on the hardware available in the industry.

1.1. Motivation

TSN network utilization is in demand in time-critical applications, i.e., automotive [8], and industrial automation spheres [9]. Nonetheless, during the TSN utilization process in industrial applications, a considerable number of challenging tasks can be faced. Performance of TSN configuration can be problematic during design, analysis, and simulation phases when considering the timing requirements of the specific application. In the design and simulation phases, detailed and complex measures upon configurations are required. Those measures can be manifested in terms of creating offline schedules for the periodically scheduled traffic or similar [10]. As TSN network configuration is currently out of interest for many researchers, the existing and available TSN network designs and TSN simulation tools are marked with a lack of automation for the purposes of the network configuration [10]. Implementing TSN networks in large-scale applications that deal with more complex traffic scenarios is deemed as an error-prone, cumbersome, and time-consuming task [10]. This complexity applies to both TSN simulation platforms and TSN hardware platforms. Therefore, an adequate automated framework is needed to facilitate the design-time and pre-simulation configurations of TSN networks [10].

If TSN is requested to be implemented, the network capability is improved in terms of bandwidth, latency, additional services, and applications. Those applications, such as voice recognition, video-on-demand services, etc., put a great demand on network resources and put challenges on hardware for processing significant volume traffic at a high-speed [11]. Timestamping on the soft-

ware level has difficulty in achieving the precision which is needed by time-triggered Ethernet protocols [12]. Consequently, hardware support is required.

Compared to the TSN hardware platforms, TSN simulation platforms enable faster development of the TSN applications. They allow changes in the TSN switch configurations via Graphical User Interfaces (GUI) and provide an abstract view of the network's structure. Whereas configuring the TSN network and TSN switches in the real-world requires thorough knowledge of the switch's specifications, which are most commonly vendor-specific and are provided as datasheets. When configuring the switch, firstly, the switch datasheet has to be revised to determine which registers need to be set up to allow a specific gate schedule. Times of gate openings are also defined in registers. If just one of the registers is not appropriately set, the entire system will not behave as expected. Some problems with wirings, power supply, component availability, CPU speed, software installations, software-hardware compatibilities, etc. could be detected while using the hardware platform for TSN utilization. Even though simulation results are a reasonable basis for all sorts of measurements and evaluations, the hardware platform needs to be included to have some expectations from a real-world environment. Hardware is used so that all the complexities mentioned can be addressed so that the research community has an insight into possible solutions and the ability to upgrade them.

To be able to prove the theory it would be the best to implement a use case that uses both Ethernet and TSN communication. Since TSN, in theory, guarantees lower latency, and in some configurations, lower jitter or no jitter at all, measurements obtained in experiment could help researchers to prove that theory holds in practice. Also use case implementation could detect possible problems when using hardware available in industry and could also provide their solutions.

1.2. Problem Formulation

As discussed in the subsections above, the configuration of the TSN network can be a challenging task. Therefore, an appropriate research problem can be addressed. Already explained scientific and engineering aspects lead to developing a functional prototype in industrial settings. The industrial use case can be based on a network model for an actual application, preferably a network in an automated construction machine, to address defined complexities when creating such a hardware system. This use case is meant to explore the effects of different configuration modes in terms of assumptions on the frame preemption and the configuration of the Credit-Based Shaper (CBS) and Time-Aware Shaper (TAS) mechanisms. The results obtained with this thesis can favor the transition from standard Ethernet to TSN network. Therefore, they could represent a basis for the future research and advancement of TSN configurations in the automotive industry. Furthermore, it is crucial to measure the variations in the overall network performance, i.e., variations in latencies and release jitter in different TSN traffic classes. These measurements should support theory and explain why TSN should be used instead of Ethernet.

Meanwhile, the thesis aims to answer the following questions:

- RQ1: What are the main benefits of replacing a conventional Ethernet network with a TSN network in an automotive use case?
- RQ2: What are the effects of network configurations on the TSN network performance with respect to response time and jitter in an automotive use case?

The answers to these research questions are obtained from the literature review and the use case implementation, which is described in the later sections.

1.3. Thesis Outline

This Thesis for the Degree of Master of Science in Computer Science with Specialization in Embedded Systems report consists of the following sections. In Section 2., main terms relevant to the research are explained and described, such as real-time systems, Ethernet, switched Ethernet, and

TSN. Section 3. presents already done related work in the field of TSN and real-time communication. In Section 4., the research methodology is justified. The description of the work is presented in Section 5., where more information about the use case, hardware setup, blob detection, switch configuration, and message transmission can be found. In Section 6., obtained results are presented, as well as the discussion about gathered measurements. The final words, conclusion, and the future work of this thesis can be found in Section 7..

2. Background

TSN set of standards is a promising solution to be applied in numerous real-time domains such as the automotive domain [1]. CAN buses are about to be replaced by automotive Ethernet in most applications, where high bandwidth is required [13]. Standard Ethernet uses protection against unnecessary bandwidth consumption, burst sizes, and malicious or improperly configured endpoints, which could apply to TSN networks. In order to design dependable real-time networks based on the TSN set of standards, various analyses in terms of timing and scheduling should be performed. For example, response-time analysis and schedulability analysis. More detailed definitions and understandings of real-time systems and networking technologies are proposed in the following sections.

2.1. Real-Time systems

An embedded system is a computing system or device intended for performing a specific function. Both hardware and software are integrated into an embedded system, and more embedded systems are combined to make an embedding system [14]. Embedded systems are usually specialized, efficient, reliable, and reactive. It can be said that embedded systems are specialized because they are commonly designed for one specific task whose functionality is not changeable. This is the main characteristic that differs embedded systems from the general-purpose system. Herewith, all excess functionalities can be taken into account as drawbacks. Over time, it has been shown that embedded systems can be energy, memory, run-time, and weight-efficient [15]. This property contributes to the long-term cost calculations. Reliability gives the information about system susceptibility to failures, and it can be calculated with specific parameters like Mean Time Between Failures (MTBF) and Mean Time to Repair (MTTR). If an embedded is requested to satisfy some critical applications, then the mentioned parameters should be of the order of minutes [15]. The reliability element should be considered in the design stage, and it depends on a product's purpose. For some systems, it is not enough to react. It is also important to react at the right time. In order to satisfy dependability and timing requirements, many embedded systems are implemented as real-time systems.

A real-time system is a system that is required to provide a logically correct response (output) at the correct times. Computation correctness depends on logical correctness and time needed for results production. If a real-time system fails to execute tasks in a defined period, the consequences could be severe for the environment and human life. A real-time system's functions are defined by a chain of real-time tasks that are coordinated to generate the output of the real-time system. The tasks may be all inside one end station or require communication with each other from different end stations through networks via messages. In case the tasks communicate through messages, the communicated message must be delivered in a defined period between real-time tasks' interactions. The predictability feature is incorporated within a real-real time system, which means that all constraints, deadlines, age, reaction response time, etc., are met with 100% certainty. Real-time systems could be distinguished according to time constraints as hard, firm, or soft [16]. If deadlines are not met by the tasks and the outcome results in catastrophe, we talk about hard real-time systems. We can find them in automobiles, airplanes, industrial control, etc. Firm real-time systems are the ones in which the tasks do not meet deadlines, but outcomes do not lead to severe consequences, e.g. database system transactions and satellite tracking. Soft real-time systems are neither hard nor firm. Suppose deadlines are not met by the tasks occasionally in a soft real-time system. In that case, the outcome can lead to a utility decrease after the expiration of the deadline, which can affect the Quality of Service (QoS).

2.2. Ethernet

Ethernet is used widely in many applications because of its cost-effectiveness, speed, scalability, and security features. Ethernet is a standard for data transmission over a local area network (LAN) [17]. LAN is a computer network that connects more devices within a limited geographical area. In general, LAN provides high-bandwidth communication, and it is enabled through low-cost

transmission media. For these reasons, Ethernet and Wi-Fi are the most typically used technologies for it[18]. In 1976, the concept of Ethernet was firstly presented by XEROX PARC [1], and in 1983 it was standardized as IEEE 802.3. Network topologies such as bus, star, line, tree, etc., are supported by Ethernet (see Fig. 1).

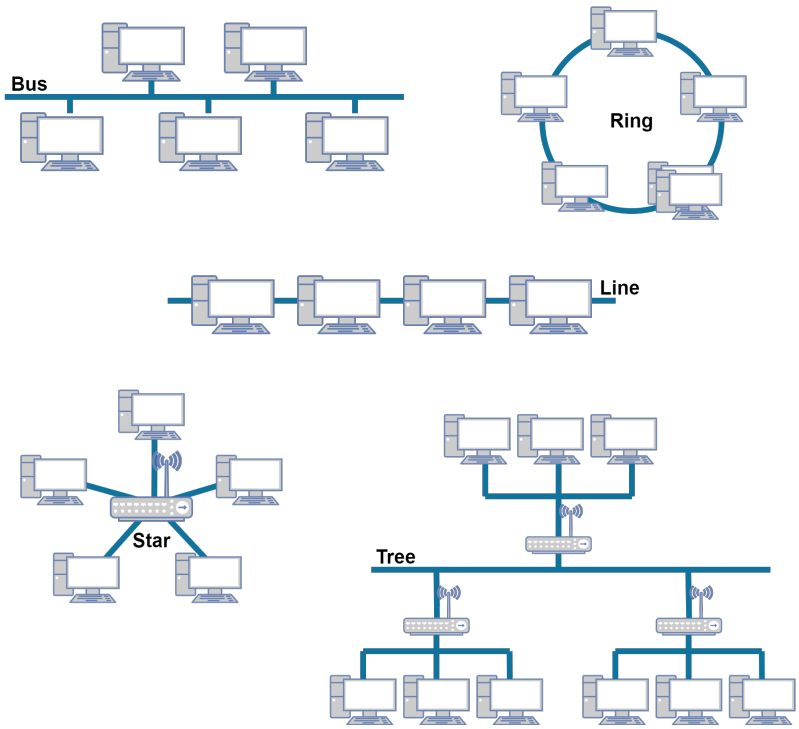

Figure 1: Types of network topologies

Ethernet fits the physical layer, and data link layer in the Open Systems Interconnection Model (OSI) reference model [17]. OSI referenced model is based on a proposal developed by the International Standards Organization (ISO), and it provides the connection of open systems - systems that are open to communicating with other systems [19]. The OSI model is shown in Fig. 2.

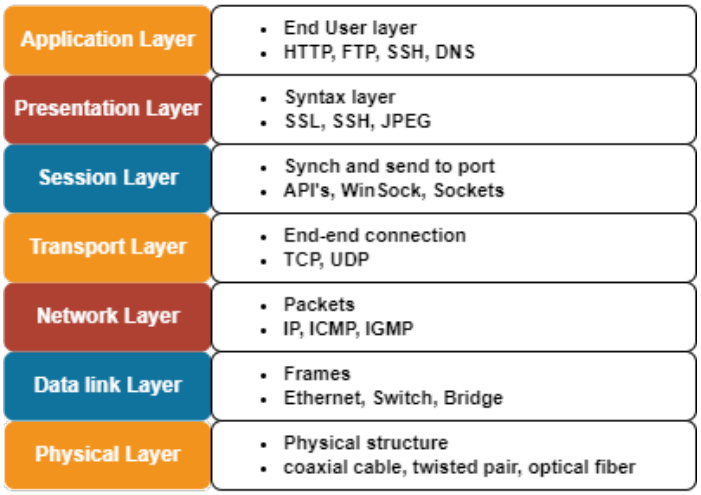

Figure 2: The OSI reference model

[1]Today known as Palo Alto Research Centre (PARC).

The speed of transmission of the data depends on the physical layer, and it can reach up to 400 Gbit/s [20]. In the physical layer, coaxial cable, twisted pair, and optical fiber are commonly used as media in Ethernet networks. Furthermore, in the data link layer, Media Access Control (MAC) sub-layer serves for network arbitration, and the Logical Link Control (LLC) sub-layer serves for flow control, synchronization, and error checking [21].

2.2..1 CSMA/CD

Carrier sense multiple access/collision detection (CSMA/CD) is an access method used by Ethernet, and it is intended for broadcasting, listening, and detecting collisions. Each node is constantly listening to the state of the network (Carrier Sense). If a network is quiet, multiple nodes can detect it and start the transmission (Multiple Access). If collisions happen, every transmission will fail, and retransmission will be established. However, frequent collisions under heavy traffic can lead to unbounded end-to-end transmission response time, which is the main disadvantage of CSMA/CD. Therefore, the arbitration mechanism can be improved by adding the data link layer device named switch in LAN. [17]. If conventional Ethernet is in use, efficiency could be exponentially decreased when more nodes want to transmit the data. For this reason, switched Ethernet brings the solution to this problem by not sending data to all nodes in the network but just to the prechosen ones. Most of today's LAN is referred to as switched Ethernet because of the achieving higher efficiency [21].

2.3. Switched Ethernet

Usage of a switch instead of a hub differs switched Ethernet from the traditional Ethernet. A hub is used for data transmission from the source to all nodes in the network, and it uses a half-duplex link. On the other hand, a switch is used for data transmission from the source to the node with the predetermined address, and the full-duplex link is in use so messages can be sent and received simultaneously by the node [22]. Each switch in the network creates a table of MAC addresses of the nodes, which are directly reachable. After that, multiple coexisting transmissions in the network are allowed for different receivers. The disadvantage of having a switch instead of the hub in the network is reflected in the latency, which is negligible but acceptable. When the switch decides to transmit the data to the receiver, the table has to be examined for the MAC address of the receiver so that the messages can be transmitted appropriately [23]. The traffic control can be controlled, and so end-to-end response times can be bounded [21]. Low jitter, cost-effectiveness, scalability, easy deployment, easy maintenance, message collision prevention, sender nodes isolation, full-duplex links, and bounded end-to-end transmission response time make switched Ethernet adequate for Audio-Video Bridging (AVB) and TSN.

2.4. Ethernet in Real-Time Communication

Higher bandwidth, the IEEE standardization, a broad range of data rates, and Internet Protocol (IP) support make automotive Ethernet the solution for replacing small networks connected with gateways with one homogeneous in-vehicle network [1]. As in the automotive industry, Ethernet is attractive to use even for the other real-time systems' applications. As previously stated, conventional Ethernet has disadvantages over guaranteed data transmission. Industrial Ethernet or Real-time Ethernet standards use modified MAC sub-layer, and they have been proposed to provide low latency and deterministic behavior overall. Ethernet-based protocols like Internet Protocol (IP), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP) are not applicable for real-time traffic. However, the hardware needed for the mentioned protocols could be used with other possible protocols [23]. Ethernet-based real-time communication standards are: EthernetPowerlink2 [2], TTEthernet [3], CC-Link IE Field4 [4], Profinet7 [5], and TCnet3 [6]. The mentioned standards can not be compatible with the other networking technologies, but TSN offers a

[2]EthernetPowerlink2: https://www.ethernet-powerlink.org/
[3]EthernetPowerlink2:https://www.tttech.com/technologies/time-triggered-ethernet/
[4]CC-Link IE Field 4: https://www.cc-link.org/en/cclink/cclinkie/cclinkie_f.html
[5]Profinet7: https://us.profinet.com/technology/profinet/
[6]TCnet3: https://www.toshiba.co.jp/sis/en/seigyo/tcnet/technology.htm

possible solution to this problem [21]. Likewise, secure data transmission and high data transfer rates up to the range of gigabytes are supported by TSN. Both talker and listener and Ethernet switches have to support TSN functions in order to get the best effect while using TSN [24]. The research based on TSN technology is legitimate and approved because of switched Ethernet, stable transmission, high bandwidth, low latency, and cost-effectiveness.

2.5. Time Sensitive Networking (TSN)

Time-Sensitive Networking (TSN) is a group of standards developed by the IEEE 802.1 TSN task group to support high-bandwidth, time-critical, and low-latency communication over the switched Ethernet [10]. The main TSN principle is controlling the flow of traffic at endpoints, so deterministic behavior could be created [21]. The TSN standards have roots in IEEE 802.1Q-2005 and IEEE AVB Task Group. First standard prioritized Best-Effort (BE) traffic with higher Quality-of-Service (QoS) properties. Double standard, AVB, introduced two new AVB classes which enable bounded latency. AVB is the forerunner of today's TSN. This standard offers several features. Three of them are taken as the most important ones:

- Common notion of time - achieved by synchronizing local clocks of switches (know as bridges in IEEE terminology) and end stations taking into consideration that they are time-aware nodes,
- Bandwidth reservation for real-time classes - presented in IEEE 802.1Qat standard; resources within the switches between the source and the destination can be reserved through Stream Reservation Protocol (SRP),
- Prioritization and traffic shaping for real-time flows - presented in IEEE 802.1Qav standard; traffic bursts are prevented through a Credit-Based Shaper (CBS) the at the output ports of switches and end nodes guaranteeing bounded latency to real-time classes [1].

On top of the AVB standard set, TSN consists of new standards providing several additional benefits and features that are especially important for applications with critical networks, which are:

- Real-time - Guaranteed latency, low-jitter, and zero congestion loss for all critical control loops,
- Convergence - Reducing complexity and costs converging networks,
- Open, Interoperable Standard - truly open standard, hence IEEE 802.1 and other relevant groups to ensure long-term viability and innovation,
- Immunity - Critical traffic is immune to effects of converged, non-critical traffic,
- Ease-of-Use - making networking easy in both design, provisioning, and maintenance [25].

2.5..1 Standards

Available TSN standards are:

- IEEE Std 802.1AS-2020 - Timing and Synchronization for Time-Sensitive Applications,
- IEEE Std 802.1CB-2017 - Frame Replication and Elimination for Reliability,
- IEEE Std 802.1CM-2018 - Time-Sensitive Networking for Fronthaul,
- IEEE Std 802.1CMde-2020 - Enhancements to Fronthaul Profiles to Support New, Fronthaul Interface, Synchronization, and Syntonization Standards,
- IEEE Std 802.1CS-2020 - Link-local Registration Protocol,
- IEEE Std 802.1Qbu-2016 - Frame Preemption,

- IEEE Std 802.1Qbv-2015 - Enhancements for Scheduled Traffic,
- IEEE Std 802.1Qca-2015 - Path Control and Reservation,
- IEEE Std 802.1Qch-2017 - Cyclic Queuing and Forwarding,
- IEEE Std 802.1Qci-2017 - Per-Stream Filtering and Policing,
- IEEE Std 802.1Qcc-2018 - SRP Enhancements and Performance Improvements,
- IEEE Std 802.1Qcp-2018 - YANG Data Model,
- IEEE Std 802.1Qcr-2020 - Asynchronous Traffic Shaping,
- IEEE Std 802.1Qcx-2020 - YANG Data Model for Connectivity Fault Management [7].

Each of the TSN standards targets one or more of the following four design aims:

- Bounded low latency
  - IEEE 802.1Qbv-2015 - Leverages on a transmission gate mechanism that is applied to the egress queues of a switch port to allow for transmitting traffic according to a predefined schedule, implemented as a list of timed gate operations that cyclically repeats,
  - IEEE 802.1Qbu-2016 in combination with IEEE 802.1Qbr - Allows for frame preemption to intersperse express (i.e., real-time) frames among best-effort ones,
  - IEEE 802.1Qch-2017 - Introduces a mechanism according to which frames are received and transmitted alternately for a fixed interval of time, called a cycle time; this way, real-time frames are accumulated during a cycle and transmitted in the following one, thus minimizing the transmission jitter and guaranteeing bounded end-to-end response times,
  - IEEE 802.1Qcr-2020 - Defines the Asynchronous Traffic Shaping (ATS) stream selection that enables the transmission of mixed real-time traffic types, such as periodic, rate-constrained, and event-driven (sporadic),
- Timing and synchronization
  - IEEE Std 802.1AS-2020 - Improves clock synchronization reliability and provides replication mechanisms to limit the probability of decreasing the synchronization accuracy in case of switch fault or frame loss,
- High reliability
  - IEEE Std 802.1CB-2017 - Deals with frame replication and elimination for reliability; by leveraging frame identification capability, it allows for duplicating frames and sending them over multiple disjoint routes in order to increase the probability that at least one of the replicas will eventually reach the final destination,
  - IEEE Std 802.1Qci-2017 - Introduces support for per-stream metering and monitoring, error detection, and error mitigation by blocking a stream or a port to enforce the error containment so that it will not propagate on the network,
- Resource management and network configuration
  - IEEE Std 802.1Qat - Defines SPR,
  - IEEE Std 802.1Qcc-2018 - Extends the capabilities of SPR and describes protocols to provide support for Configurable Stream Reservation (CSR) classes and stream [1].

[7] Time-Sensitive Networking (TSN) Task Group: https://1.ieee802.org/tsn/

2.5..2 Types of Traffic

The transmission selection algorithm in the IEEE 802.1Q standard defines two classes, critical and non-critical ones. Critical traffic classes are Classes A, the higher priority one, and B. Non-Critical class is defined as the best-effort (BE) class with the lowest priority. The scheduled traffic (ST) class is defined in the standard with strict temporal isolation via the TAS mechanism [37]. This method works by implementing a gate mechanism that prevents traffic from accessing a queue until it is permitted, according to the gate control list (GCL) that repeats periodically. Each queue in a TSN switch has its gate configuration to achieve temporal isolation. There can be up to eight different classes in each port that can be open or closed; the queue with the highest priority is drained first [21].

2.5..3 Scheduling Mechanisms

Available TSN standards as a feature propose various traffic shapers, such as Time-Aware Shaper (TAS) standardized by IEEE 802.1Qbv, Credit-Based Shaper (CBS) standardized by IEEE 802.1Qav, and Asynchronous Traffic Shaper (ATS) standardized by IEEE 802.1Qcr [26]. These shapers are most commonly used for flow control and execution of traffic shaping, that is, the scheduling. Since presented shapers are unique and can be used separately and in combination, it is difficult for the end-users to choose the right shaper for their application.

TAS is used for prerun-time scheduling of deterministic time-triggered traffic and preemption of the frames. Frame preemption is essential in stopping the ongoing frame transmission in favor of an urgent case frame. Thereby, latency and jitter reduction will be the outcome for time-triggered frames. There can be found up to 8 queues on the egress port of the switch. Between each queue and the transmission selection, a so-called gate is placed. To enable time-triggered communication, TAS relies on the global network clock used to control the gate of each queue on a predefined schedule [26]. This schedule is also known as a gate-control list (GCL) that repeats itself cyclically. Each gate can be defined as either closed or open. If the gate is opened, the frames can be forwarded through the egress port. Frames that are being forwarded are sorted in each queue according to another scheduling algorithm. One of the simplest algorithms states that the one with the highest priority is released among all open queues with a pending packet [27].

CBS represents an asynchronous bandwidth management shaper that sets the amount of bandwidth for AVB traffic. In other words, it is used for critical classes (A and B) for starvation prevention of low-priority critical traffic [37]. Classes A and B correspond to a FIFO queue, and they have their credit value that the CBS shaper uses to govern AVB frame transmission [26]. CBS is sometimes assigned a private queue between a gate and a queue of egress ports. The frame can be sent only if the credit value of the queue is non-negative. If the frames are waiting inside the queue, the credit value rises with an “idle slope” rate. If they are being sent through the queue, the credit value decreases with the “send slope” rate. Credit is set to zero if credit has a positive value, but there is no frame pending for transmission. The explained method prevents lower priorities from being annulled over a more extended period [27].

3. Related Work

Through time embedded software and systems have enabled rapid development of the automotive industry. Rapid development and functionality advancements of considered systems resulted in complex software solutions dealing with the problem of fulfilling strict timing requirements. As a possible solution to some of the problems, the TSN task group has been proposed as an extension to IEEE 802.1 Ethernet. With the TSN standard, high bandwidth and low latency requirements are addressed. Compared to Ethernet, TSN also considers safety and security requirements that allow its usage in safety-critical applications. In work, Ashjaei et al. (2021) [1] debate recent developments in the field of automotive embedded systems that impose the adoption of TSN and give an overview of the current usage of TSN in automotive applications. Several features have been singled out that set TSN apart from standard Ethernet. Those are possible support of reliable clock synchronization, the ability to reserve resources for the different traffic types, the ability to apply traffic prioritization and traffic shaping, and overall efficiency. They agreed that the best way to investigate the potential of TSN in the automotive domain thoroughly is to make it part of the new development processes of automotive embedded systems. In the future, they plan to focus on TSN sub-standard, IEEE 802.1Qcr-2020, as it proposes bounded latency asynchronous shaping with robustness properties.

It is always hard for a person who proposes a solution based on a TSN to choose an appropriate standard that will be the best fit for the application that is taken into consideration. Zhao et al. (2021) in [26] try to provide information that will help researchers with the selection of appropriate TSN sub-standards for the use cases. These standards introduce various traffic shapers such as TAS, ATS, CBS, and Strict Priority (SP). To analyze the quantitative performance of individual and combined traffic shapers, many experiments were done. Upper bound of delay, backlog and jitter were one of the parameters that were considered. To their knowledge, they were the first to do this kind of quantitative analysis. They concluded that SP shaper is more beneficial to high-priority traffic transmission delay than CBS. At the same time, CBS specifies bandwidth reservations for each priority traffic. When compared to SP and CBS, ATS positively affects low priority traffic, while it has some limitations when considering high-priority traffic. Only if high-priority traffic in the network reaches around 80% of the average traffic load than it can show its superiority. TAS shaper that implemented flow-scheduling of Time-Triggered (TT) traffic showed the highest performance with ultra-low latency, jitter, and backlog. The only problem that arises when considering a large number of flows in networks is a synthesis of optimized GCLs. A combination of different traffic shapers in the same architecture is proposed as a possible solution to this problem.

Performance study in [37] can also help researchers to understand how different configurations implicate the performance of TSN networks. In this work, unexplored configurations that include TAS and CBS together with frame preemption were discussed. Frame preemption, in this case, helps to reduce latency and jitter of high priority frames by stopping the transmission of an ongoing frame in favor of an urgent one. Critical traffic classes A and B undergo the CBS mechanism to prevent starvation of low-priority critical traffic, while ST traffic undergoes strict temporal isolation via the TAS mechanism. Suppose ST class is not present in the network, then TAS is not used. So considered configurations are divided into ones that have TAS enabled and those that have it disabled. It is concluded that when TAS is enabled, and A and B classes are set as express, ST classes experience a blocking delay. If ST classes are set to express, they experience blocking delay, which is bounded by the maximum size of one frame, whereas if they are set as preemptable, blocking delay can be longer depending on the number of express frames in the transmission queue. If TAS is disabled, then the class that is set as express will have improved response times as long as lower priority classes are not set to express. Since available tools that simulate TSN do not support configurations that combine TAS, CBS, and frame preemption, this study also proposed a C/C++ simulation platform that was developed by the writers and can be easily integrated into NeSTIiNg or CoRE4INET.

In addition to analytical techniques for schedulability analysis, TSN simulation and modeling

tools can be helpful for more efficient configuration design and scheduling of TSN networks [1]. OMNeT++ [8] is one of open-source simulation platforms primarily used for network simulation. INET2 [9] is another tool that is used for modeling wired, wireless, and mobile networks. INET4 [10] is a extension to INET2. Its main benefits are the more powerful multidimensional representation of radio signals and support for TSN infrastructures. Core4INET [28] and NeSTiNg [29] simulation frameworks are built on OMNeT++ [2]. NeSTiNg simulation framework supports several TSN features, including the scheduled traffic (IEEE 802.1Qbv), frame preemption (IEEE Std 802.1Qbu and IEEE Std 802.3br), credit-based shaper (IEEE Std 802.1Qav), and time synchronization (IEEE Std 802.1AS) [**?**].

Mubeen et al. (2021) in [30] provided an additional offline schedule optimization solution that configures TAS for ST classes to achieve high QoS of lower priority BE traffic. They propose three alternative objective functions, namely Maximization, Sparse and Evenly Sparse, followed by a set of constraints on ST streams. After conducting experiments in NeSTiNg, it is shown that proposed functions outperform the state-of-the-art ST scheduling solutions that focus on minimizing the offsets of the ST classes. Not only the given solution affects positively on BE classes, but it is believed that it can also affect A and B classes.

Since TSN is relatively new, there are not so many hardware and software tools that support its implementation. For this reason, Krumacker et al. (2021) [27] decided to design and implement their own TSN mechanism based on a network simulation framework. The developed solution is highly flexible and can be used in various use cases. They decided to use the ns-3 simulator since it allows them to achieve an implementation that will fit into its modular framework.

It was already mentioned in [26] that large networks demand complex design of configurations. So if researchers decide to configure large networks, they would need to do that manually since existing frameworks support only manual configurations of TSN. Houtan et al. (2021) decided to design a modular framework that will automate the process of offline scheduling. They used available state-of-the-art simulation frameworks for demonstration and evaluation purposes, mostly NeSTiNg. The designed framework allows automatic translation of TSN flow schedules without manual intervention. Usage of the proposed framework saves the user from error-prone, time-consuming, and challenging manual configurations. In the future, the plan is to integrate this framework into existing industrial tools that use TSN network communication.

To overcome challenges posed by the rising number of network participants, usage of the Automotive Open System Architecture (AUTOSAR) adaptive software platform is proposed in [31]. Combined with Open Platform Communications United Architecture (OPC UA), a communication standard popular in modern industrial automation, PubSub, an architecture that promotes scalability and works in conjunction with TSN, provides deterministic high-speed communication.

[8] OMNeT++: https://omnetpp.org/
[9] INET2:https://inet.omnetpp.org/
[10] INET4:https://inet.omnetpp.org/

4. Method

Experimental study methods allow the experimenter to understand better and evaluate the factors that influence a particular system using statistical approaches. These kinds of approaches combine theoretical knowledge of experimental designs and a working knowledge of the particular factors to be studied [32]. Each experiment is conducted in several steps, as shown in Fig. 3. These steps can be done in few iterations depending whether the results are good or bad. If they are good, conclusion can be made, if not than independent variables are again defined and steps 2, 3, 4 and 5 are repeated. In addition to the previously defined steps, each experimenter must generate a research question, state a hypothesis, and define variables. A proper experimental design relies on the relationships testing between and among variables. In general, one variable, meaning the independent variable, is controlled in order to measure its effect on other dependent variables [33]. The focus of this thesis work are the effects of various configuration modes [23] on the overall performance of the network in an automotive use case.

In this specific experiment, we evaluate the network performance of a vehicular use case - taking into account jitter and delay as performance metrics. The change of dependent variables, jitter and delay, is the result of experimental manipulation of independent variables defined in the use-case study that is going to be used. Egress port configuration consists of several independent variables such as gate-control list, clock cycles, port delays, etc. The use-case is based on a network scenario for an actual application, preferably a network in an automated construction vehicle. The use case - is selected to represent a typical automotive use case and has been previously prepared at Volvo CE. Alongside the on-site experiments, we also take advantage of simulation tools at the configuration time and before implementing the use-case on the actual hardware. This would allow us to understand our targeted network metrics better. Moreover, the research is done experimentally on the actual hardware by setting up the use-case in industrial settings on-site at Volvo CE, configuring the network, performing experiments, and evaluating the measurements' extracted data.

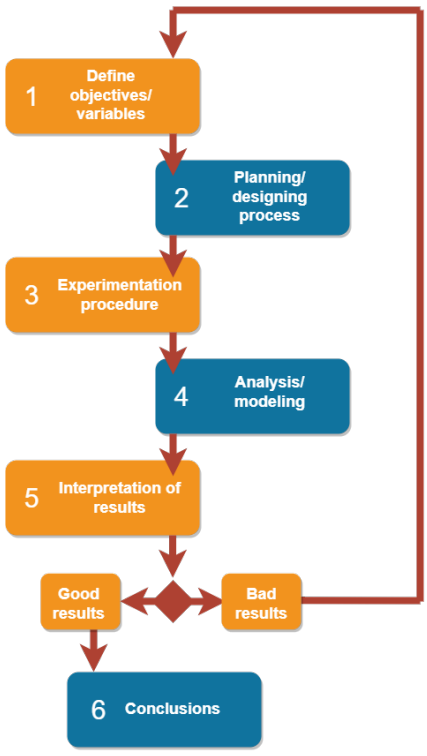

Figure 3: The steps of experimental research applied on system under study

Implementation of an experiment is just one part of the research process. The whole research work can be divided into the following steps, such as:

1. Literature review;
2. Review of hardware;
3. Configuration;
4. Implementation; and
5. Evaluation (see Fig. 4).

If obtained results in evaluation step are not good enough, than independent variables need to be defined again, which means that switch needs to be configured from the beginning. Thereby, whole process is considered to be done in few iterations. This is the reason why researcher can go back from step 5 to step 3 in Fig. 4.

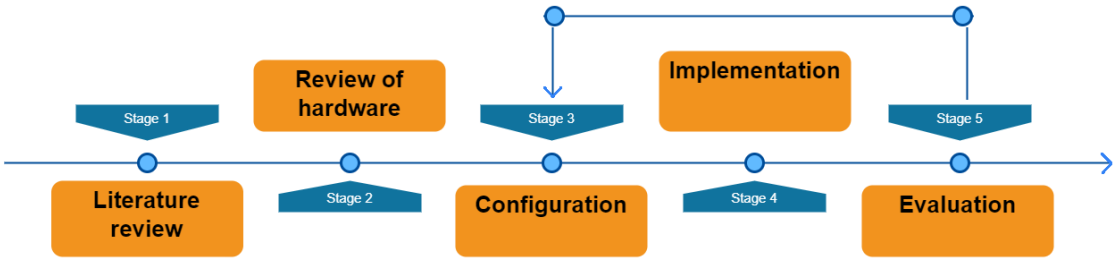

Figure 4: Research steps

5. Description of the work

Before the experimental setup, a literature study is performed on theories behind TSN and the state-of-the-art use of TSN. In this section, the idea and the process of the project are about to be described. Accordingly, the next phase is related to constructing a vision and the system on which the experiment can be performed. The discussion about the use case implementation is presented in the later phase. The end-to-end latency evaluation is carried on after the use case realization with the appropriate hardware and software tools. It is important to point out that due to the company's guidelines and rules, the details about hardware and codes [11] will not be presented in the thesis.

5.1. Vision

As day by day passes, technology and science evolve, and the industry has to keep up with the challenges. The processing of large amounts of data in real-time systems questions whether hardware and software platforms can stand up to the challenge. It is necessary to examine TSN network performance and compare it to the conventional Ethernet performance. When it comes to real-time communication applications like vehicular systems, TSN is a promising communication solution. TSN is primarily new, so TSN-supported simulation and hardware environments are not widely available. Hence, the authors of this work have decided to implement a single-use case based on TSN communication technology to inspect the advantages and disadvantages of replacing the conventional Ethernet network with TSN. The designed hardware system needs to mimic the one already implemented in today's vehicles.

The vehicular system's network is consisted of around 100 ECUs, and they can perform engine, brake, transmission control, etc. [34]. With other additional components, ECUs and switches together make a real-time system. If TSN is requested to be implemented and the network performance to be measured, then the system has to consider its total capacity. The complexity arises with the reached number of 100 million source lines of code (SLOC), which is equal to over 1 GB of the software code number of lines of code [1]. This complex network can be simulated in one of the simulation tools like OMNeT++ or NeSTiNg. However, hardware implementation in a real-world environment is more complex and challenging to deploy. The authors of this work consider the general use case presented in Fig. 5. As mentioned before, the goal is to have all real-time nodes.

To implement the idea of making this kind of a system, available hardware, including real-time operating ECUs and TSN supported switches, should be at the disposal. Every Ethernet switch needs to have options for implementing TSN standards to enable scheduling mechanisms like CBS and TAS. Hypothetically, if the system is constructed with all of the components from the Fig. 5, it would be difficult to add or remove one of the TSN flows. It is recommended to have the same switches in the system since one of the TSN´s disadvantages is the complexity of the configuration. The switch can be configured on the lower level by setting registers and ports with the information obtained from datasheets. If one of the registers of the switch is not set correctly, TSN advantages can not come to the fore. These problems are not present in simulation tools. In general, problems detected in simulation tools are easier to deal with than those in real-hardware platforms. The collection of the simulation results is accessible in the same tool simulation was implemented. On the other hand, obtained results from real hardware platforms are challenging to access, and additional software and hardware need to be included. This leads to new problems and obstacles to the final goal.

No matter how the implementation may seem complex, the focus has to be put on the TSN advantages, such as low latency and high bandwidth, compared to the conventional Ethernet network. Consequently, a simpler version of the system can be implemented to get the needed information about hardware setup and to evaluate end-to-end latencies and jitter.

[11] Exception is code for blob detection.

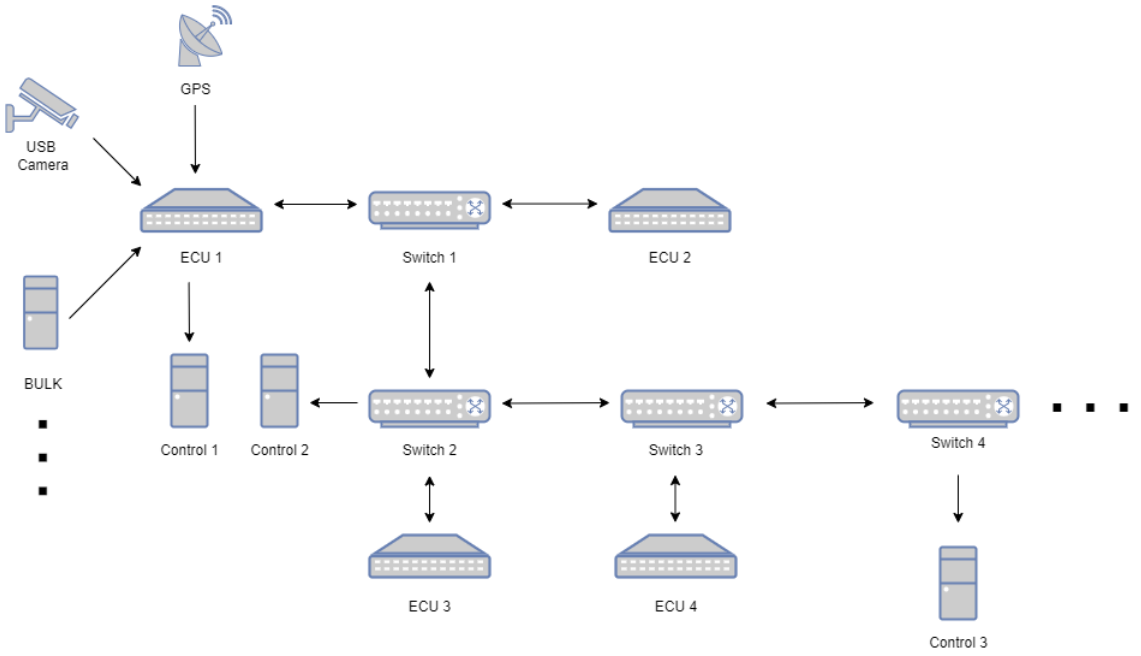

Figure 5: Starting vision of the use case scenario

5.2. Use Case Description

As proof of the mentioned concept in the previous section, the demonstrator can be developed in the industrial settings at Volvo CE, one of the vehicle providers in the construction vehicle domain. When considering current industrial in-vehicle platforms, it is essential to mention that neither one of the nodes is in real-time. This opposes the vision that we had. The communication is reduced from one node to another. End-to-end communication incorporated with TSN offers real-time properties of the network. It is crucial to analyze timings with the assumption of having a defined workload and delays. With the newer functionalities brought up with the technologies like artificial intelligence and machine learning, the nodes are obligated to do more calculations than they do now. It would be demanded to process 100 MGB of data under microseconds in the upcoming years. Therefore, a problem might arise regarding the timing constraints for end-to-end communication. These advanced technologies demand better timing investigation and higher bandwidth. Wherefore, the authors of this thesis decided to test TSN communication by incorporating one of the complex operations like image processing. For this reason, data might be generated from only one source, and the network can be additionally congested by producing some bulk. Messages sent through the TSN switch are generated in an app that detects color blobs. Since none of the available ECU's operating systems can support installing and using applications needed for more complex operations, such as the one mentioned, we need to use a PC. It is essential to note that the available PC's operating system is Windows. However, it is possible to run codes from Virtual Box and XUbuntu. So, ECU 1 in Fig. 5 might be replaced with a PC and media converter.

At this moment, there are only two identical Ethernet switches available in the Volvo industrial environment that enable TSN features and standards. These switches only support Qav and Qbv, meaning CBS and TAS can be enabled, respectively. One switch is connected to the two ECUs on the mutual hardware platform. One ECU is Rubus-based, and one is Android-based. Through the text, this hardware platform is referred to as SECU. Only one SECU will be considered for simplicity's sake. Android-based ECU is not used in work.

Once a vision for developing such a system was proposed in the section above, the investigation of real-time properties was conducted in the industrial environment. Therefore, it has been seen that there are many engineering challenges. Because of the difficulties defined, it is optimal to

simplify the desired system to the one with the devices mentioned before (see Fig. 6). The usage and evaluation of simple use cases are beneficial due to the fast error detection and more accurate measurements.

In the following, a simple use case is considered for implementing TSN network configuration. The system consists of a USB camera, PC, switch, ECU, and signalizing light. The idea is to write a code for blob detection while the camera is turned on and send messages through TSN to the switch and ECU. A message is a simple form of zeros or ones depending on if the blob is detected. The ECU can be connected to a signalizing light. The signalizing light can represent a real-world indication of potential disturbance or danger. In this way, the briefly explained small system mimics complex real-world systems and accompanying processes. Essential parts: the data source, primary communication, and control parts are included in the system. Through the next part of the thesis, evaluation and discussion of results acquired from the system in Fig. 6 are conducted.

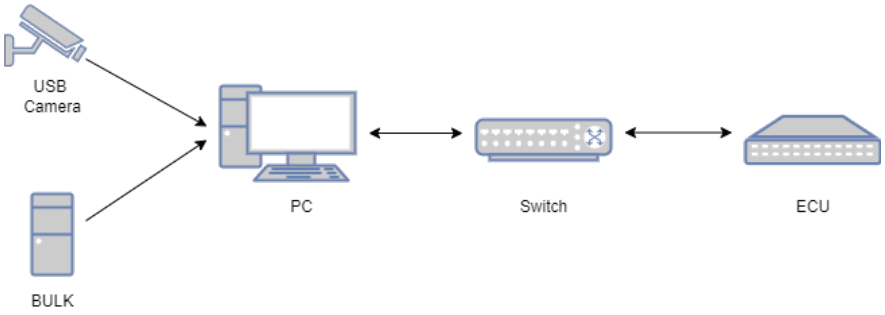

Figure 6: The use case network topology

A simple USB camera is placed for video streaming and blob detection. The camera is connected to the PC, from which simple messages will are sent to the switch and ECU. Besides the data coming from the camera, a bulk producer can be placed additionally. The purpose of the bulk traffic node is to occupy the link between the PC and the Ethernet switch. Thereupon, the message can be sent to the final destination. The bulk traffic is generated without knowing the actual size and the period of the packages in time. The intent is to observe whether all packets are coming from the sender to the receiver despite having bulk. The ability to provide deterministic behavior over Ethernet despite the traffic load is going to be shown as a property related to TSN.

Obtaining results from existing industrial tools for modeling automotive embedded systems and providing TSN configuration is the action taken with this thesis. The switch, already set to the industrial settings, has to be examined for AVB/TSN support. As the purpose is to have a system as operable and straightforward as possible, TAS scheduling is chosen instead of CBS. Therefore, the traffic in the network is limited to ST and other traffic for the rest of the bandwidth. Qbv standard must be included when selecting adequate registers and properties because TAS is chosen. After the appropriate system configuration and testing, more ECUs and switches can be included in the network.

A two different configurations on the same network topology shown in Fig. 6 can be inspected. They are described as:

1. Ethernet configuration
2. TSN configuration - Qbv standard

Each of the configurations is observed with no frame preemption. The intention is to demonstrate a use case under Ethernet and TSN scenarios and to examine results in terms of response time and jitter.

5.3. Hardware setup

Before writing adequate codes for blob detection, switch configuration, and message transmission, an appropriate hardware setup is needed. Hardware components are gathered according to the use case description, explained in 5.2.. A detailed reading of datasheets of the required hardware components is needed. One of the most challenging parts of this thesis is the switch configuration. The steps to be taken in the configuration process are manual setup and low-level programming. More details about it are written in 5.5.. Thus, it is important to show that TSN can be enabled on the industrial Ethernet switch currently in use. If this happens to be impossible, the economic cost of implementing new advancements could be increased. Therefore, an Ethernet switch and ECU are chosen as commonly used ones in the automotive industry and as the ones that are currently only available. The switch is already configured to the Ethernet, and in that way, it can be used for getting the measurements for later comparison. The ethernet switch is chosen because TSN is an Ethernet-based set of standards.

The ECU is connected to the switch on a mutual platform, and they are both connected to the power supply of 12 V and 5 A through an appropriate connector. Likewise, the switch and ECU are connected to the outer connection board needed for linking up with the debugger. The debugger is requested to download the configuration code to the microprocessor and read the parameters. The parameters' reading proceeds after writing code for the switch configuration because before sending frames with TSN, it is essential to check registers' values with the debugger. After the parameters have been successfully read, the process can be continued. Processes of debugging and code downloading are done using the appropriate software tool installed on a PC because the debugger is connected directly to the PC via a USB cable.

Since PC is suitable for handling complex operations like image processing, it is placed instead of ECU for use case realization. First, the PC's operating system was Windows with installed Virtual Box and Xubuntu. Linux is an open-source operating system, and Windows is a commercial one. Therefore, privacy could be an issue for Windows, where data is not protected as it is in Linux. In some versions of Linux, data can be military-grade encrypted, so the information is not possessed by anyone except the official user. One of the disadvantages of Windows can be manifested in its firewall, which monitors incoming and upcoming traffic. Linux firewall is better than Windows because Linux uses the Netfilter Linux kernel subsystem, and Windows uses a program running in the user node. Linux firewalls have greater control over network connections and more advanced features. When experiment started, Xubuntu in Virtual Box was too slow while running blob detection app. To prevent additional delays and potential problems when starting a camera, the authors of this work decided to use the Windows operating system.

A USB camera is attached to the second PC for color blob detection.+ If the video resolution is chosen as 1080 pixels, thirty frames per second are sent from the camera. On the other hand, if the video resolution is selected as 720 pixels, then sixty frames per second are sent from the camera. For this project, full HD frames are sent, which means that frames are sent with the frequency of 30Hz. A media converter is needed between the PC and the switch for sending messages so TSN communication can be established successfully. Media converters allow the interconnection of two different cabling technologies. In Ethernet systems, most commonly copper (twisted-pair) segment is connected with the fiber optic segment [35].

As an output of the overall system, signalizing light, whose state identifies whether the blob is detected or not, is used for the demonstration. Signalizing light represents a simple LED placed on the ECU, which is a part of the mutual hardware platform (the one where the Ethernet switch and ECU are installed). The code for controlling LED is incorporated within the same project as the code that reads a signal value from the frame that came to the ECU port.

To be able to calculate time constraints, one more media converter is connected between the ECU and the third PC, which reads the messages that are being sent through the frames in the

app that tracks the traffic. The app is called Wireshark [12]. Overall, three PCs are being used. The first one that downloads and debugs the code on a switch can be left out, and those operations can be done on any of the other PCs present in the designed system. Because of the simplicity and not wanting to burden other PCs that measure times, it is better to use all three PCs. A more detailed hardware schema is presented in Fig. 7.

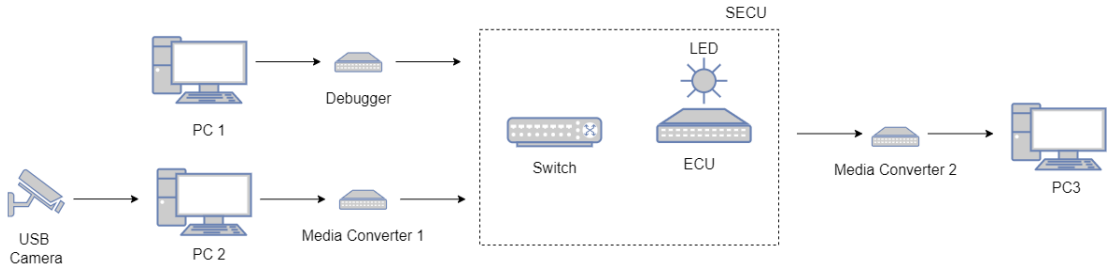

Figure 7: Detailed hardware schema

5.4. Blob Detection

Before the designed use-case can send TSN messages, those messages need to be generated somehow. An app that is recording a video using a USB camera and detects a blob of defined color is generating the desired message depending on whether the color object is detected.

For a PC video represents an array of frames where each frame represents an image. When a user is looking at the video on a PC, she/he is looking at several images at one moment. The images change in frequency, which is invisible to the human eye. Every image can be seen as a unique information carrier. Several image processing techniques can be done to identify and detect a single feature or an object of interest. For implementing the software solution in this use case, the OpenCV library is used in the python environment. OpenCV is a highly optimized open-source library that supports computer vision and machine learning, focusing on real-time applications.

As a color of interest in this use case, blue is chosen. The best way to identify objects of a certain color is to threshold the image's pixel values in a defined color space. If appropriate threshold values are chosen, then it is certain that just objects in that color-space range are highlighted. Threshold values also depend on the color space in which the image is represented. Most popular color-spaces are RGB (*R- red, G - green, B - blue*) and HSV (*H - hue, S - saturation, V - value*). The authors of this work decided to use HSV color space because in the Fig. 8 very similar shades of green color can be seen and have completely different values in RGB color space.

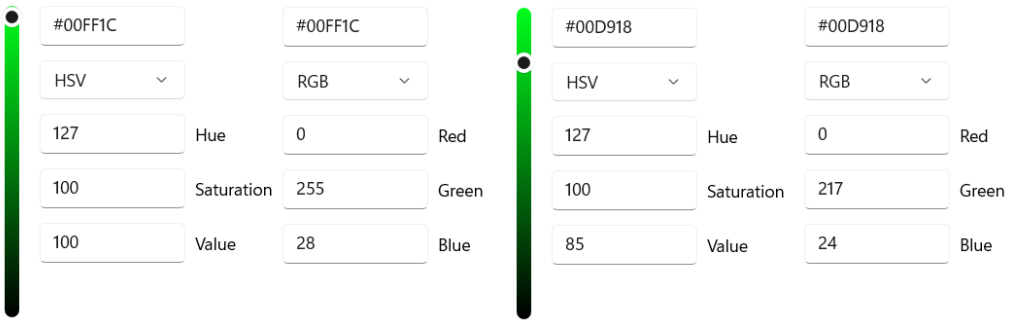

Figure 8: RGB and HSV values of slightly different green color in Microsoft Paint

This is why the image is transferred to HSV color-space when the multi-spectral threshold method is used. The final thing left is to define the range of the H, S, and V for blue color.

[12]Wireshark: https://www.wireshark.org/

This can be done by visual inspection or using some software tools[13]. Lower HSV (represented as (H,S,V)) boundary for blue color is taken as (100,90,36) while upper boundary is taken as (130,255,255). Each value ranges from 0 to 255 since each pixel represents an 8-bit value. H, S, V boundaries can be additionally changed using the sliders on the bottom of the Fig. 9. Black and white image on the Fig. 9 is a result given after applying multi-spectral threshold on Fig. 10.

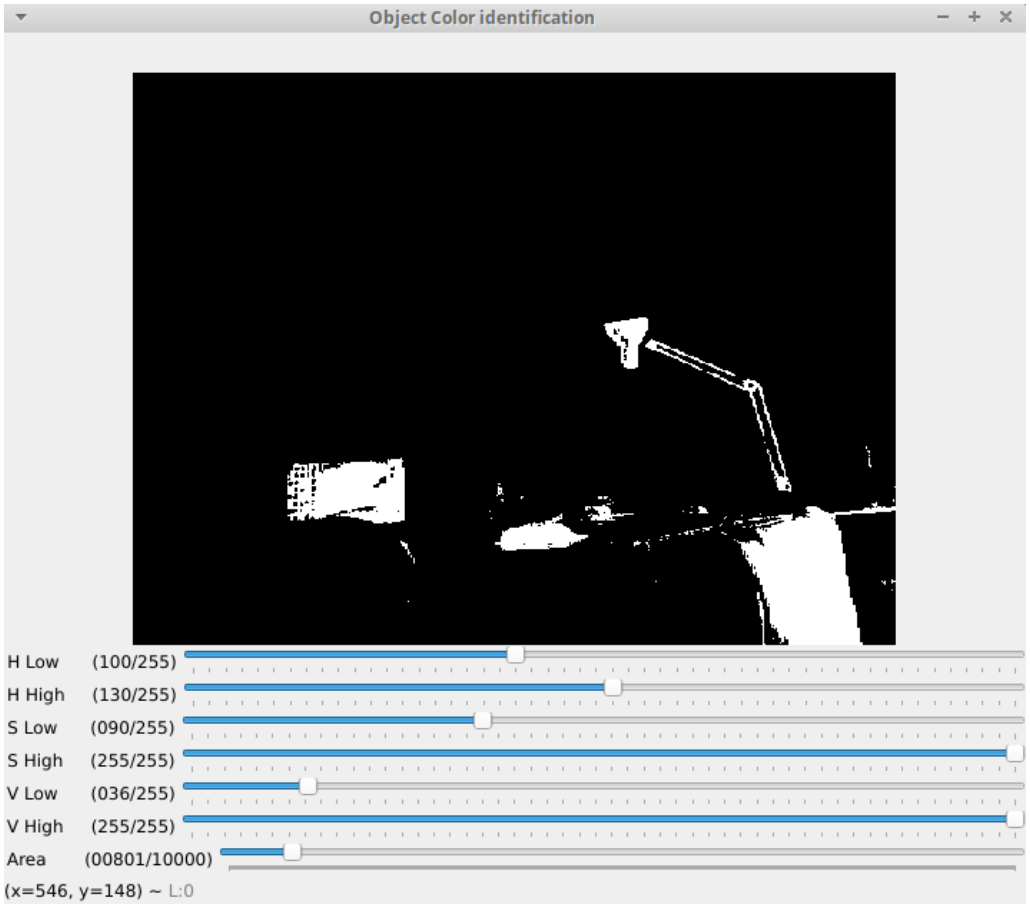

Figure 9: Result of using multi-spectral threshold in HSV color-space

After all blue objects have been classified, the next logical step is to identify the contours of each detected object somehow so that a rectangle surrounding that object can be drawn. Sometimes some of the identified contours can represent a false-positive result, and sometimes some of them cannot be recognized as a unity. These are the hidden problems that lay behind this interpretation. In order to minimize the error of the first-mentioned problem, two solutions are presented in the following. One of them is identifying just one blue object, which is the one with maximum area. The problem with this solution is that sometimes several blue objects must be identified. Then the second solution should be used. In the second solution, every object with an area smaller than a minimum area value is neglected. After applying all previously defined steps, it can be said that blue blobs are successfully detected. Final result is given on Fig. 10.

[13] i.e. Microsoft Paint

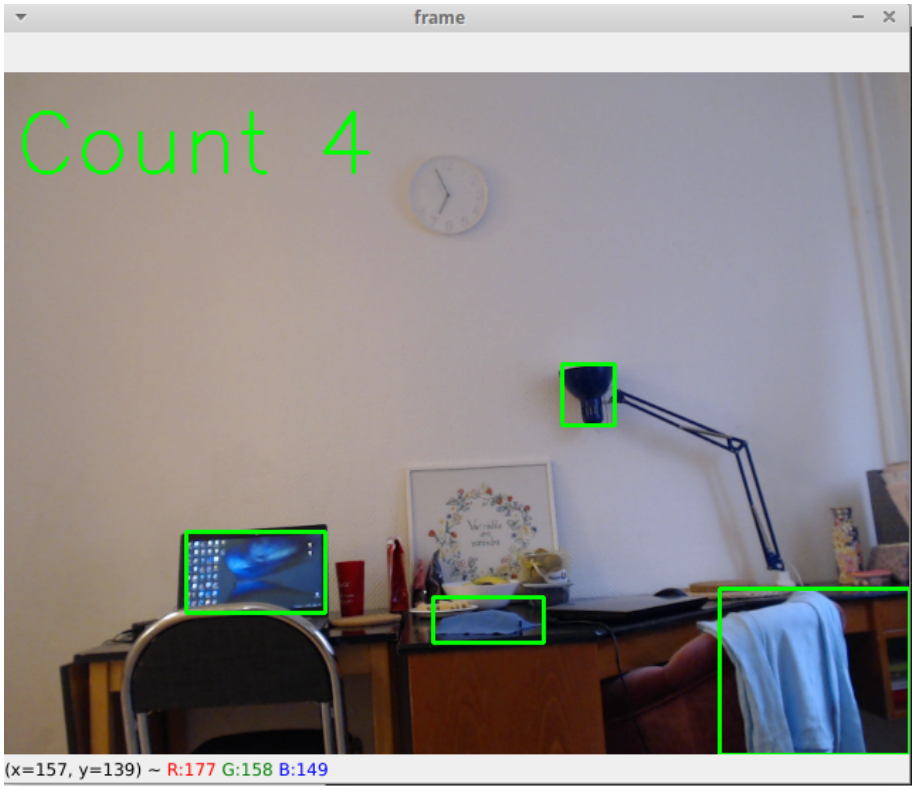

Figure 10: Result of blob detection app

5.5. Switch Configuration

Before hardware setup is explained in 5.3., documentation of the switch that is in use has to be read through to examine if AVB/TSN could be enabled. After a detailed examination and identification of the possibilities, it was concluded that AVB/TSN reconfiguration of the switch is possible using Qbv or Qav standard. However, the preemption feature is not included in the settings list. Thereby, it is not considered in the experimental study. As the chosen switch is embedded in the company's products, the procedure of the Ethernet configuration has already been established. The configuration code is written in C++ in Visual Studio. As TSN is an Ethernet-based set of standards, only a few modifications are needed to set appropriate registers for AVB/TSN. The switch device contains global registers, affecting all AVB and TSN functions. These registers can be accessed using the AVB command register and the AVB data register. Offsets of the global register for AVB command and data register are needed to be set to continue the configuration. AVB data register has a field for 16 bits of data, which indicates whether the data is read or written, depending on the AVB command register. AVB command register is divided into the bit fields: busy unit, operation code unit, port, block, and address.

The global AVB registers are used to access AVB blocks with the specific values for 802.1AS Precision Time Protocol (PTP) and Time Application Interface (TAI) registers, 802.1BA Audio Video Bridging (AVB/TSN) Policy registers, 802.1Qav per Class Shaping and Pacing registers, 802.1Qbv per Queue Time-Aware Shaper registers. Since TAS is chosen, the Qbv feature needs to be enabled. Consequently, Qbv registers for delay times for each time window, port table control, and table control must be set. GCL is set differently depending on the configuration that we consider. It is going to be further explained in Section 6.. Two time delays with two different entry sets can be chosen. The maximum resolution is 64 ns of delay per count, so the range is between 0 and 4.194 ms. Port table control refers to the pointer and the queue's state. The pointer points to the desired Qbv port table to what data to read or write. The index registers are accessed for entry sets and guard bands with the pointer bits. For the Qbv and guard band

entries, queue state bits indicate which egress queues on a port are allowed to participate in the selection for the next frame during the transmission. At most, 16 entries can be set. In the table entry register, window time and guard band are fixed. In increments of 8ns, the window time is fixed, so the maximum time is 262.136 μs. Alongside, selection of the time sets is proposed with the Qbv control register.

5.6. Message Transmission

As part of the use case realization, messages have to be transmitted from the PC through the switch to the ECU. As the PC's operating system is Windows, and for better performance and the presence of the company's firewalls, it was demanded to install Virtual Box and Xubuntu. A Linux-based environment is commonly used among programmers to establish communication because Linux offers better security and reliability. The writing of Python code and installing OpenCV is not as complex as it could be by using Windows. After doing some testing and obtaining some results, it was concluded that Xubuntu is too slow for blob detection app and that available PCs cannot support this virtual operating system in its full capacity. To prevent additional delays and to have better performance of PCs Windows operating system was used at the end.

Two frames, which are being sent through the switch to the ECU, are generated in two separated .py scripts. There are two types of frames. The first frame has the highest priority and carries the information of the detected blob, while the second frame has the lowest priority and represents the bulk traffic. Information carrier frames are generated in the blob detection script. They carry 13 bits of data where two of them contain information about the ordinal number of the frame (in the range of 0-255), one bit that signalizes whether the blob is detected or not, and six bits that represent the time when the frame is sent (see Fig. 11). These frames are sent every 50 ms. UDP protocol is chosen above TCP on the application level for this type of frame due to the company's guidelines.

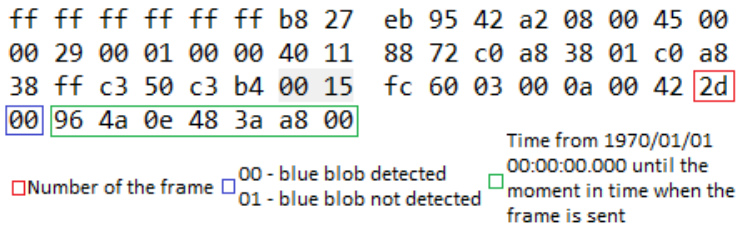

Figure 11: Sample of a high-priority frame sent from the PC2

The bulk frames are generated to congest the network with the help of the Internet Control Message Protocol (ICMP). Network devices can use ICMP for sending error messages and information, which indicates failure or success of communication to the other IP address. In an Internet Protocol version 4 (IPv4) packet, the ICMP packet is encapsulated, and it consists of header and data sections (see Fig. 12). The maximum length of ICMP error messages is 576 bytes [36]. Both Python scripts are run at the same time. The bulk framers are sent one after the other without specific sleep time.

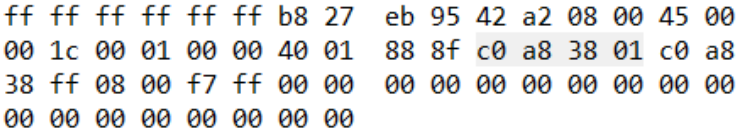

Figure 12: Sample of a low-priority bulk frame sent from the PC2

As one of the research questions of this thesis is associated with network performance measurements, timestamps added when the message is sent and when it is received are considered. The

difference between those two times represents a delay. If the delays are not consistent and there are some oscillations in the values, then that occurrence is called jitter. In theory, jitter is downsized with the usage of TSN.

A software tool that monitors the traffic on both PC2 and PC3 Wireshark [14] will be used. Wireshark is an open-source tool intended for analyzing network protocols. Wireshark provides information about frame number, time from the 1970/01/01 00:00:00.0000 until the moment in time when the frame is received (time format can be adjusted), source and destination IP address, the protocol that frame uses (in this case UDP or ICMP), length, and other helpful information (see Fig. 13). Also, by selecting one of the preview frames, it is possible to see every bit of a frame and its meaning. Before generating traffic, it is important to have the same subnet mask set for the sender PC and the receiver PC. By this, the packets from the sender can be sent to the receiver, which is detecting them.

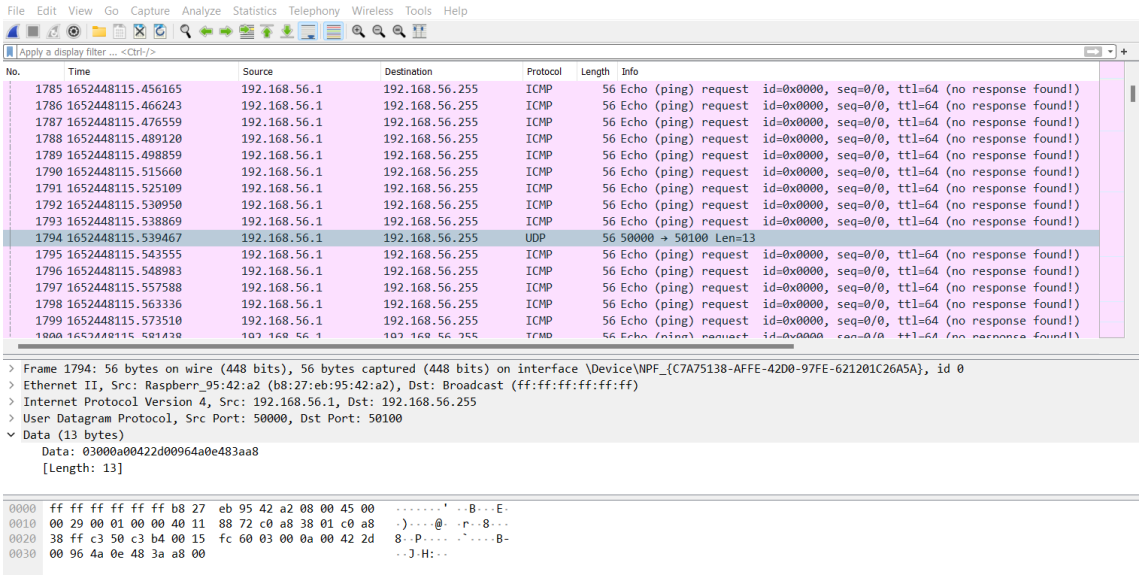

Figure 13: Wireshark interface

The existing literature restricts the TSN network configurations by two assumptions: the priority of the scheduled traffic (ST) class is higher than other classes' priorities, and the ST class is express while other classes are preemptable. A preemptable class can be preempted by the express class, which can not be preempted by any other high-priority express class [37]. Frame that carries information is marked as ST traffic class. Since Qbv standard is used it implicates the usage of TAS scheduler. This scheduler guarantees that all time-critical traffic classes, in this case high priority ST frames, will arrive exactly on time without any jitter by granting exclusive access to transmission flows.

5.7. Limitations

Almost all limitations were mentioned and described in more detail in the sections above. In this section, just a short overview is given.

A physical demonstrator, including electronic control units (ECUs) and cables, was built based on the use case. Different configuration modes were investigated depending on the physical switch capabilities in the used ECU. Therefore, some features, e.g., preemption, may not be feasible to include in the experiment. In those cases, an alternative switch may also be used. Since the switch, together with two other ECUs, is integrated into SECU, it cannot directly communicate with the

[14] Wireshark: https://www.wireshark.org/

ECU of the author's choice. This limits the possibility of choosing an ECU with a different operating system or different features. Refresh time of integrated Rubus-based ECU can vary between 5 and 20 milliseconds. This may represent a problem in applications where an instant response is needed.

The need for OpenCV in combination with Python or C/C++ for detecting a blob of the desired color is a limitation of another kind. Since there is no available ECU running on a Real-Time Operating System (RTOS) that supports OpenCV, at least one node of a network needs to be a non-real-time node. This changes the nature of the proposed vision. PC that sends the frames and PC that receives them does not support TSN communication, so a mediator between the PCs and switch two media converters are used. This is one more limitation of having a PC instead of an ECU. Both PCs run on Windows operating system, but Linux is preferred because of its reliability, security, and easier usage.

6. Industrial configuration of the use case

The first step in setting up a use case was an examination of hardware available in industrial settings of Volvo CE. In addition to existing hardware, an HD camera suitable for the use case was bought. Before connecting all nodes and components, the authors of this thesis had to exhaustively examine the datasheet to enable one of the TSN standards on the Ethernet switch. As it was possible to enable appropriate registers for Qav or Qbv, for simplicity's sake, Qbv was chosen for activating the TAS scheduling mechanism. As previously said, TAS was elected as a result of putting into consideration just one high and one low priority class. Qbv leverages a transmission gate mechanism applied to the egress queues of a switch port to allow for transmitting traffic according to a predefined schedule, implemented as a list of timed gate operations that periodically reruns. The next step was writing a .cpp and .hpp scripts containing functions that set up the correct values in a register. In the same script, an additional function that reads the registers' values was also written to ensure that all registers were set appropriately. The second step was connecting all network nodes so that PCs that send and receive the frames are able to communicate. On both PCs, Wireshark is started, so all message logs are recorded. The traffic was recorded for a couple of minutes while sending information about when the blob was detected and when not. In order to have answers to the research questions presented in Section 1., measurements obtained from use case realization need to be extracted and evaluated. The measurements are presented in this section.

6.1. Comparative evaluation of measured end-to-end latencies

For stability and correct functionality of software operations ensurement, end-to-end latencies are significant [38]. A time interval needed for a frame to be transmitted from a sender to a receiver is called end-to-end latency [10]. End-to-end latencies could vary, and therefore they are calculated for each sent frame. Latency is considered as the difference between the packet receiving time and the packet sending time. Both moments in a specific time are recorded on the receiver PC and sender PC by Wireshark. Additionally, jitter is the change in latency [39]. Jitter is calculated as the absolute difference between two consecutive delays. End-user devices and applications are designed to accept the tolerable amount of jitter by buffering the data flow and compensating for small latency changes. Therefore, information about latency and jitter is essential when characterizing the network performance over time [39].

Exploring TSN network configuration on a real hardware platform offers the opportunity for further advancements in the realization of the general use case explained in 5.1.. In this subsection, measurements gained from two scenarios are presented. In the first scenario switch is configured to Ethernet, and in the second to TSN. MATLAB tool is used for the purpose of graphical representation of latency and jitter values in Ethernet and TSN configurations.

6.1..1 Ethernet configuration

The Ethernet configuration of the chosen industrial switch has already been tested in other projects at the company. Thereby, the first step is to obtain results with a switch configured to the Ethernet. By conducting this experiment, it would be shown if there are any problems with the hardware, establishing the appropriate connection, and expected results. The latency values are shown in 15. The jitter values obtained with Ethernet configuration are shown in Fig. 16 and Fig. 17. The maximum and minimum values of latencies and jitter, as well as average jitter value, are shown in Table 1.

6.1..2 TSN configuration

After getting results from the Ethernet configuration scenario, the next step is to check the TSN configuration. The TSN configuration code is downloaded to the switch, and the process is repeated. TAS scheduling mechanism is enabled with Qbv registers on the industrial Ethernet switch,

and block diagram for it is shown in Fig. 14. In the first period, only highest priority gate is opened while all others are closed. In the second one, all gates are opened at the same time. The sleep time is set to 50 ms.

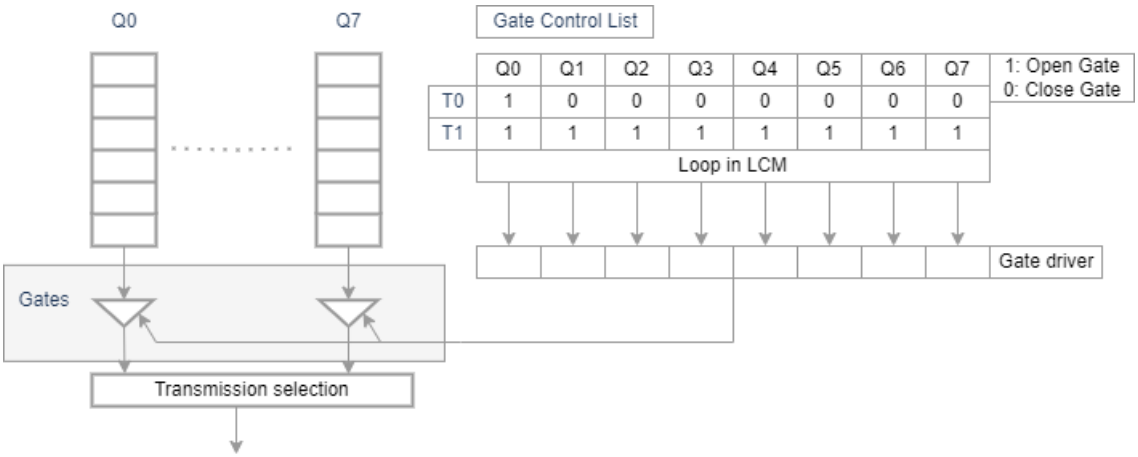

Figure 14: Time-Aware Shaper (TAS) block diagram

Graphs on Fig. 15, 16 and 17 are result of analyzing 429 frames. Number of frame is places on y-axis. On x-axis latency, jitter and average jitter values are placed. The latency values are shown in Fig. 15. Obtained results are in seconds. Since frames are being sent every 50 ms, it was expected that measured latencies are in range of milliseconds, not in seconds. TSN latencies are marked in red colour, and Ethernet latencies are marked in blue. TSN latencies seem to be constant over time, while Ethernet ones are decreasing. This decrease in Ethernet latencies can be consequence of the clock drift.

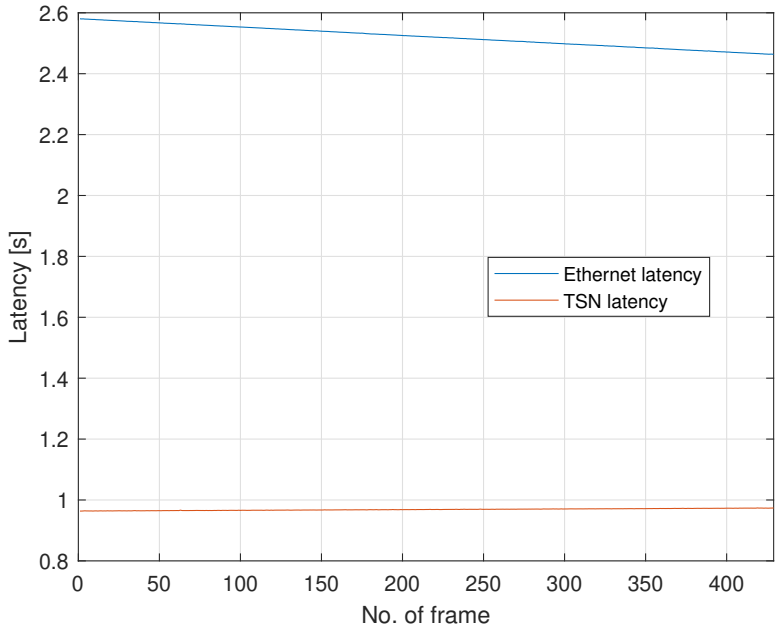

Figure 15: Comparison of Ethernet and TSN latencies

The jitter values obtained with TSN configuration are shown in the Fig. 16 and Fig. 17. Jitter values are obtained in miliseconds. Values corresponding to TSN are marked in red, and ones corresponding to Ethernet are marked in blue. Since it is hard to make any conclusion when

analyzing Fig. 16, average value of jitter was calculated and presented on Fig. 17. On both graphs it can be seen that TSN jitter values are approximately two times less that Ethernet ones. A detailed discussion of the results obtained on these graphs will be presented in 6.2..

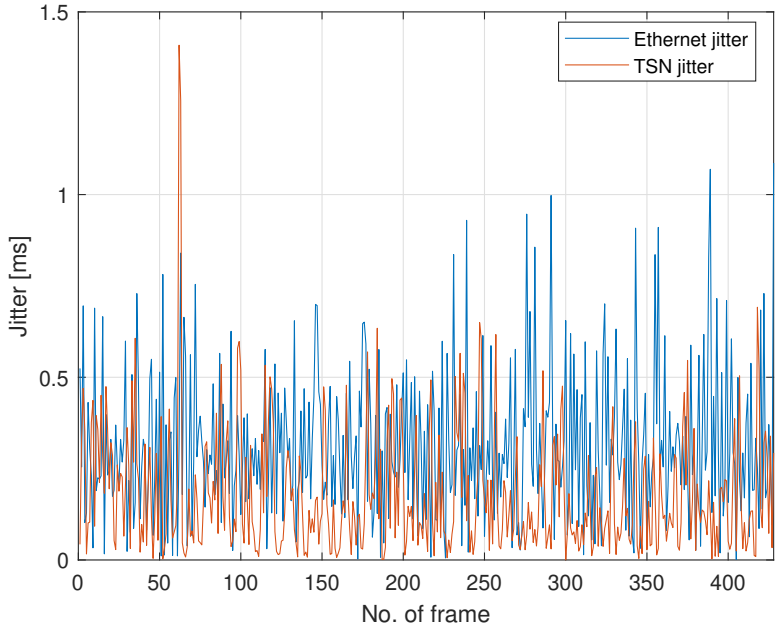

Figure 16: Comparison of Ethernet and TSN jitter

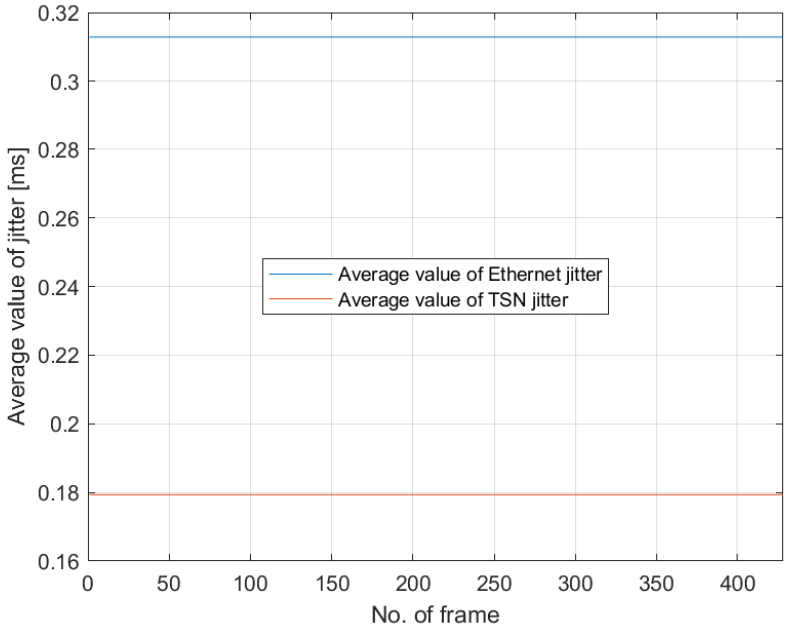

Figure 17: Comparison of Ethernet and TSN average jitter

Table 1 is comparing measurements for Ethernet and TSN. It presents the maximum and minimum values of latencies and jitter, as well as average jitter values. This table provides numerical values which are easier to compare when analyzing which communication has better performance

and whether obtained results are in satisfying ranges.

Configuration	Max.latency [s]	Min. latency [s]	Max. jitter [ms]	Min. jitter [ms]	Avg. jitter [ms]
Ethernet	2.58032	2.45383	1.08647	$1.43051\text{x}10^{-3}$	0.31286
TSN	0.97387	0.96353	1.40882	$2.384186\text{x}10^{-4}$	0.17932

Table 1: Minimum and maximum latency and jitter values obtained from measurements

6.2. Discussion

In this subsection, the results related to the use case explained in 5. are analyzed and discussed. Furthermore, the critical points for additional improvements to the project and usability of TSN configuration are shown.

Understanding latency and jitter in network performance is essential in comparing communication technologies and upgrading systems for future demands, i.e., faster data processing. In Fig. 15, it could be seen that Ethernet latency slowly decreases as the number of sent frames increases. On the other hand, TSN latency stays approximately constant over time. The latency was measured in seconds, which could be the result of the clock drift or timestamping. Even though measured latencies are not in a expected range, it is clear that in each case constant system delay is present, but still delay in case of TSN is constant over time and is approximately two times less than Ethernet one. Jitter is accumulated more in the case when Ethernet was configurated, and it was graphically presented on Fig. 16. The order of measured jitter is in milliseconds. Only problem is that jitter should not be present when considering ST traffic class scheduled by TAS in TSN communication. Usage of Windows instead of Linux, measuring times when frame was sent and received in Whireshark, instead of analyzing timestamps of frames, are possible sources of jitter. This is why low jitter presence is justified and does not affect conclusions made when comparing TSN with Ethernet.

This thesis's aim is to examine the possibilities of TSN configuration in industrial settings and measure network performance in terms of latency and jitter. The configuration of the TSN network can be a challenging task. Already explained scientific and engineering aspects have led to developing a functional prototype in industrial settings. The industrial use case was based on a network model for detecting color blobs and measuring network performance. This use case was meant to explore the effects of Ethernet and configuration modes in terms of assumptions on the TAS scheduling mechanism. The results obtained with this thesis are in favor to the the transition from standard Ethernet to TSN network. As shown in theory, the latency and jitter were proved to be smaller for TSN configuration than for standard Ethernet configuration. Therefore, this could represent a basis for further research and advancement of TSN configurations in the automotive industry. The considered use case had several limitations. More nodes could be added to the network, making it more complex. The usability of the current switch configuration to TSN could be extended for additional switches. Therefore, if the switches, the network would be, to a greater extent, more similar to the one present in modern vehicles. Reaching the final goal of replacing conventional Ethernet with TSN has become closer for a step further.

In the following subsections, it is discussed more about the key improvement areas of this experimental study.

6.2..1 Improvement area 1 - Precision of the measurements

Keeping in mind that the settings in Wireshark were set to show the time passed from 1970/01/01 00:00:00.0000 until the moment in time when the frame is received, the number of digits and decimals was too big to be presented in the .csv file. The .csv file was needed for MATLAB representation of latency and jitter graphs and also for easier data manipulation. Therefore, the first five or six decimals were only taken for further calculations. Timestamping on the software

level has difficulty in achieving the precision which is needed by time-triggered Ethernet protocols [12].

6.2..2 Improvement area 2 - Increase of the number of included traffic classes and usage of other scheduling mechanisms

As it was already discussed, because of simplicity and because only two traffic classes were considered TAS mechanism was used to avoid the complexities of CBS. The only problem that arises when considering a large number of flows in networks using TAS is a synthesis of optimized GCLs. A combination of different traffic shapers in the same architecture is proposed as a possible solution to this problem. To test how available hardware reacts when CBS or a combination of CBS and TAS are used, AVB traffic classes, classes A and B should be added. As a future work, a use case containing a larger number of switches can be designed. Large-scale networks can use different TSN standards depending on the required mechanisms for a specific task. When considering these kinds of networks configuration of switches together with designing offline schedules for considered classes presents a real challenge. If behavior of used switches is investigated when using just one of the mechanisms, and their combination, at least configuration challenges are decreased.

6.2..3 Improvement area 3 - Hardware setup

This improvement area refers to the precision of measurements area, too. During the hardware setup, multiple problems were met. As xUbuntu is installed within Virtual box, additional files had to be installed regarding Visual Studio, Opencv, drivers for camera, libraries included in .py scripts, etc. While the data was transmitted from the camera a few times, the PC was slowed down and did not recognize the camera as a device. With these and other problems related to hardware-software compatibility, the process of experimenting was slowed down.
Likewise, the PCs which were in use during the experimentation had Windows and Virtual box installed. The frames were sent from the sender PC's Virtual box to the receiver PC's Windows. Hereby, jitter could be accumulated because of the large amount of data passing through the network and cause the congestion. If the older hardware equipment was in use, i.e., the worn put cables, then it could be possible that it is not designed for handling large amounts of traffic. If the Virtual box has adequately allocated resources, but they can not be provided in time, then problems with network jitter could affect the performance of audio/video streaming [40]. Furthermore, ECU's CPU refresh time could affect on receiving messages and LED control. Herewith, the efficiency and better results could be obtained by exempting and changing some things in the hardware setup. Moreover, considering of replacing PC and media converters with ECU could lead to lower latency and jitter. Consequently, the unnecessary jitter would not affect the measurement process.

6.2..4 Improvement area 4 - Isolation of computation delays from communication delays

Limitations that obstructed us to isolate communication from computation are as fallows:

- Whireshark - Whireshark detects all frames, not just ones that are important for analysis. This is the reason why there is unexplained jitter when evaluating latencies of ST traffic classes.
- Operating system - When Windows is used on a PC that sends frames containing information a lot of applications can run in the background and interfere with frame priority.
- Timestamping frames - Frames that were sent from PC are timestamped when created. Problem is that they can not be timestamped when received on Rubus-based ECU because of the complex implementation of APIs and not being able to access it at all. This is the reason why Wireshark had to be used.

All these limitations are reason why jitter is present when considering TSN communication and why delay is represented in seconds, not milliseconds. Authors of this work tried to convert timestamps

extracted from frames and compare them with times obtained in both Wiresharks. Conversion of only ten timestamps and its comparison took more time than it was expected so this procedure was marked as unnecessary. If Linux OS is used instead of Windows, some background applications can be shut down and just blob detection app can be the one that sends the frames with the assigned priority. This can prevent other frames from being detected in Wireshark and can minimize jitter obtained in TSN communication. Also measurement procedure would be less time-consuming if researchers could access logs of the messages. Switches with graphical user interface and user friendly configuration could help to overcome these issues.

7. Conclusions

This thesis aimed to examine TSN configurations on network performance implemented with a real hardware platform. Before creating the use case, the requisite background knowledge in real-time systems, Ethernet, TSN, and scheduling mechanisms were presented. After the background section was introduced to the reader, the related works done on TSN, switch configuration, and scheduling mechanisms were presented.

Within this thesis, the future vision of TSN implementation in the real-world system has been shown. As TSN is a relatively new set of standards in the industry, not many configuration solutions are available. However, a use case on a smaller scale was created in order to show TSN advantages on network performance in terms of latencies and jitter. The use case was realized in a few steps: hardware set up, writing codes for color blob detection, message transmission, bulk creation, and switch configuration. The timing measurements were obtained from two scenarios. In the first, the switch was configured to the Ethernet and the second to TSN.

The research questions proposed in this thesis in Section 1. are answered through the discussion about obtained results in Section 6.2.. The highlights of the discussion were possible improvement areas in measurement precision, an increase in the number of included traffic classes, and usage of other scheduling mechanisms and hardware setup. Complexities of designing such a system were also addressed through the design process together with limitations. Results obtained from measuring network performance were evaluated, and theoretical assumptions were therefore proved in industrial settings. Science and theory helped authors to better understand the system and when something is not configured the way it should be, because then obtained results did not support theoretical assumptions. Industrial approach to the problem gave a different perspective to the authors and helped them realize that theory doesn't always hold in practice and that theory is powerful tool in hands of engineers who know how and when to use it. That is how this work contributes both to science and industry. Without science, industrial implementation would be impossible. Given theoretical and numerical analysis can be further improved based on knowledge gained in this work.

7.1. Future Work

The results made from this experimental study will be taken into account as this thesis is a part of a more significant project at Volvo CE. With this work, it has been shown that TSN configurations are possible with the usage of tools and equipment in the current use. The simple use case explained in 5. can be upgraded by adding more nodes into the network, e.g., more switches and ECUs. The configuration of the newly added switches would be approximately the same as the one used in this experimental study. If more nodes are included in the network, the use case will represent more of a real-world in-vehicle network. Due to the recognized limitations and obstacles during the realization of the experimental study, additional changes could be made. The PCs, which were used during the experimentation, were too slow at some moments. A more powerful PC with a better processor and bigger memory space could be requested. However, the final goal is to replace PC with ECU and get all real-time nodes with ECUs operating on a real-time operating system. In that manner, new obstacles could be met in the future. Used switches did not support preemptions, and because of that, other TSN switches could be considered if needed for other purposes. Furthermore, instead of using TAS, CBS could be assessed by considering more traffic classes such as A and B traffic classes.

It has been predicted that TSN will be dominant over industrial Ethernet because of its scalability feature. Consequently, the research area in this field is expanding, leading to an increased number of conducted experiments and analyses.

Acknowledgements

We are extremely grateful to our supervisors Sara Andersson and Bahar Houtan for their guidance and support. A special mention to Mohammad Ashjaei at the Mälardalen University and Alexander Karlsson at Volvo CE, who assisted and encouraged us throughout this thesis. Additionally, we would like to thank to the all employees at Volvo CE, who helped us to solve problems, that arose during the experimental study realization.

References

[1] M. Ashjaei, L. L. Bello, M. Daneshtalab, G. Patti, S. Saponara, and S. Mubeen, "Time-Sensitive Networking in automotive embedded systems: State of the art and research opportunities," *Journal of Systems Architecture*, vol. 117, p. 102137, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762121001028

[2] B. Houtan, "Configuring and Analysing TSN Networks Considering Low-priority Traffic," December 2021. [Online]. Available: http://www.es.mdh.se/publications/6346-

[3] A. S. Tanenbaum and M. van Steen, *Distributed Systems: Principles and Paradigms*, 2nd ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2007. [Online]. Available: https://www.bibsonomy.org/bibtex/2aeb78dfa5b91b66dce027f86ecdeeb81/flint63

[4] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Translating Timing Constraints during Vehicular Distributed Embedded Systems Development," in *1st International Workshop on Model-Driven Engineering for Component-Based Software Systems*, September 2014. [Online]. Available: http://www.es.mdh.se/publications/3656-

[5] S. Mubeen, T. Nolte, M. Sjödin, J. Lundbäck, and K.-L. Lundbäck, "Supporting timing analysis of vehicular embedded systems through the refinement of timing constraints," *Softw. Syst. Model.*, vol. 18, 2019. [Online]. Available: https://link.springer.com/article/10.1007/s10270-017-0579-8#citeas

[6] "Timing Augmented Description Language (TADL2) syntax, semantics, metamodel," *Ver. 2, Deliverable 11*, August 2012.

[7] S. Mubeen, E. Lisova, and A. V. Feljan, "Timing Predictability and Security in Safety-critical Industrial Cyber-physical Systems: A Position Paper," *Applied Sciences–Special Issue "Emerging Paradigms and Architectures for Industry 4.0 Applications"*, vol. 10, no. 3125, pp. 1–17, April 2020. [Online]. Available: https://www.researchgate.net/publication/341125958_Timing_Predictability_and_Security_in_Safety-Critical_Industrial_Cyber-Physical_Systems_A_Position_Paper

[8] L. L. Bello, R. Mariani, S. Mubeen, and S. Saponara, "Recent advances and trends in on-board embedded and networked automotive systems," in *IEEE Transactions on Industrial Informatics*, vol. 1, 2019. [Online]. Available: https://www.researchgate.net/publication/328767294_Recent_Advances_and_Trends_in_On-Board_Embedded_and_Networked_Automotive_Systems

[9] L. L. Bello and W. Steiner, "A Perspective on IEEE Time-Sensitive Networking for Industrial Communication and Automation System," in *Proceedings of the IEEE,*, vol. 1, 2019, pp. 771–778. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8695835

[10] B. Houtan, A. Bergström, M. Ashjaei, M. Daneshtalab, M. Sjödin, and S. Mubeen, "An Automated Configuration Framework for TSN Networks," in *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, vol. 1, 2021, pp. 771–778. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9453628

[11] X. Deng, Y. Yang, and S. M. Sangjin Hong, "A Flexible Platform for Hardware-Aware Network Experiments and a Case Study on Wireless Network Coding," *IEEE/ACM Transactions on Networking*, vol. 21, no. 1, 2013. [Online]. Available: https://www.researchgate.net/publication/224137006_A_Flexible_Platform_for_Hardware-Aware_Network_Experiments_and_a_Case_Study_on_Wireless_Network_Coding

[12] Y. Seol, D. Hyeon, J. Min, M. Kim, and J. Paek, "Timely Survey of Time-Sensitive Networking: Past and Future Directions," *IEEE Access*, vol. 9, pp. 142 506–142 527, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9576720

[13] M. Sauerwald, "CAN bus, Ethernet, or FPD-Link: Which is best for automotive communications?" *Analog Design Journal*, 2014. [Online]. Available: https://www.ti.com/lit/an/slyt560/slyt560.pdf?ts=1652249472748

[14] Q. Li and C. Yao, *Real-Time Concepts for Embedded Systems*, 2003. [Online]. Available: https://books.google.se/books?id=RzP3iH0BSW0C&printsec=frontcover&hl=sv&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[15] Shibu K V, *Introduction to Embedded Systems*, 2009. [Online]. Available: https://books.google.se/books?id=8hfn4gwR90MC&lpg=PA72&dq=embedded%20systems%20characteristics&pg=PR3#v=onepage&q=embedded%20systems%20characteristics&f=false

[16] K. Shin and P. Ramanathan, "Real-time computing: A new discipline of computer science and engineering," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 6–24, 1994. [Online]. Available: https://ieeexplore.ieee.org/document/259423

[17] Z. Ullah, "Use of Ethernet Technology in Computer Network," *Global Journal of Computer Science and Technology Network, Web Security*, vol. 12, 2012. [Online]. Available: https://core.ac.uk/download/pdf/231149472.pdf

[18] D. Clark, K. Pogran, and D. Reed, "An introduction to local area networks," *Proceedings of the IEEE*, vol. 66, pp. 1497 – 1517, 12 1978. [Online]. Available: https://groups.csail.mit.edu/ana/Publications/PubPDFs/An%20Introduction%20to%20Local%20Area%20Networks.pdf

[19] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*. Pearson, 2007. [Online]. Available: https://www.mbit.edu.in/wp-content/uploads/2020/05/Computer-Networks-5th-Edition.pdf

[20] IEEE p802.3bs standard. [Online]. Available: https://www.ieee802.org/3/400GSG/email/msg01519.html

[21] A. Bergström, "Automatic Generation of Simulated Time Sensitive Networking Network Configuration," Master's thesis, Mälardalen University, 6 2020. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1438082/FULLTEXT01.pdf

[22] K.-C. Lee and S. Lee, "Performance evaluation of switched Ethernet for real-time industrial communications," *Computer Standards Interfaces*, vol. 24, pp. 411–423, 11 2002. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0920548902000703

[23] H. Suljić and M. Muminović, "Performance study and analysis of Time Sensitive Networking," Master's thesis, Mälardalen University, 6 2019. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1324903&dswid=-782

[24] T. Brand, "Time Sensitive Networks: Real-Time Ethernet," 2018. [Online]. Available: https://www.analog.com/ru/technical-articles/a87000-time-sensitive-networks-real-time-ethernet.html

[25] G. A. Ditzel and P. Didier, "Time sensitive network (TSN) protocols and use in EtherNet/IP systems," *in 2015 ODVA Industry Conference 17th Annual Meeting*, 2015. [Online]. Available: https://docplayer.net/27360475-Time-sensitive-network-tsn-protocols-and-use-in-ethernet-ip-systems.html

[26] L. Zhao, P. Pop, and S. Steinhorst, "Quantitative Performance Comparison of Various Traffic Shapers in Time-Sensitive Networking," *ArXiv*, vol. abs/2103.13424, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9788573

[27] D. Krummacker and L. Wendling, "TSN Simulation: Time-Aware Shaper implemented in ns-3," 12 2020. [Online]. Available: https://www.researchgate.net/publication/348431501_TSN_Simulation_Time-Aware_Shaper_implemented_in_ns-3

[28] J. Jiang, Y. Li, S. Hong, A. Xu, and K. Wang, "A Time-sensitive Networking (TSN) Simulation Model Based on OMNET++," in *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2018, pp. 643–648. [Online]. Available: https://ieeexplore.ieee.org/document/8484302

[29] J. Falk, D. Hellmanns, B. Carabelli, N. Nayak, F. Dürr, S. Kehrer, and K. Rothermel, "NeSTiNg: Simulating IEEE Time-sensitive Networking (TSN) in OMNeT++," in *2019 International Conference on Networked Systems (NetSys)*, 2019, pp. 1–8. [Online]. Available: https://www.researchgate.net/publication/330753982_NeSTiNg_Simulating_IEEE_Time-sensitive_Networking_TSN_in_OMNeT

[30] B. Houtan, M. Ashjaei, M. Daneshtalab, M. Sjödin, and S. Mubeen, "Synthesising Schedules to Improve QoS of Best-Effort Traffic in TSN Networks," in *29th International Conference on Real-Time Networks and Systems*, 2021, p. 68–77. [Online]. Available: https://doi.org/10.1145/3453417.3453423

[31] A. Arestova, M. Martin, K.-S. J. Hielscher, and R. German, "A Service-Oriented Real-Time Communication Scheme for AUTOSAR Adaptive Using OPC UA and Time-Sensitive Networking," *Sensors*, vol. 21, no. 7, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/7/2337

[32] G. Hanrahan, J. Zhu, S. Gibani, and D. Patil, "Chemometrics and Statistics: Experimental Design," in *Encyclopedia of Analytical Science (Second Edition)*, second edition ed., P. Worsfold, A. Townshend, and C. Poole, Eds. Oxford: Elsevier, 2005, pp. 8–13. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B0123693977000790

[33] S. Bell, "Experimental Design," in *International Encyclopedia of Human Geography*, R. Kitchin and N. Thrift, Eds. Oxford: Elsevier, 2009, pp. 672–675. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080449104004314

[34] W. Choi, H. J. Jo, S. Woo, J. Y. Chun, J. Park, and D. H. Lee, "Identifying ECUs Using Inimitable Characteristics of Signals in Controller Area Networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 4757–4770, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8303766

[35] G. Thomas, "Incorporating Media Converters," *The Extension, A Technical Suplement to Control Network*, vol. 7, 2006. [Online]. Available: https://www.ccontrols.com/pdf/Extv7n6.pdf

[36] S. C. Forouzan, Behrouz A; Fegan, "Data Communications And Networking," pp. 621–630, 2007. [Online]. Available: http://eti2506.elimu.net/Introduction/Books/Data%20Communications%20and%20Networking%20By%20Behrouz%20A.Forouzan.pdf

[37] M. Ashjaei, L. Murselović, and S. Mubeen, "Implications of Various Preemption Configurations in TSN Networks," *IEEE Embedded Systems Letters*, vol. 14, no. 1, pp. 39–42, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9508365

[38] K.-h. C. J.-j. C. Marco Durr, Georg Von Der Bruggen, "End-to-End Timing Analysis of Sporadic Cause-Effect Chains in Distributed Systems," *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, 2019. [Online]. Available: https://ls12-www.cs.tu-dortmund.de/daes/media/documents/publications/downloads/2019_cases_marco.pdf

[39] I. Spirent Communications, "Measuring Jitter Accurately," 2007. [Online]. Available: https://piazza.com/class_profile/get_resource/hnd936blk4c26r/hnwio4j6bb5nm

[40] C.-L. W. Luwei Cheng and S. D, "Defeating Network Jitter for Virtual Machines," *Fourth IEEE International Conference on Utility and Cloud Computing*, 2011. [Online]. Available: https://i.cs.hku.hk/~clwang/papers/ucc2011_submitted.pdf