Politecnico di Torino
Master's Degree in Computer Engineering

Master thesis

Automotive Ethernet and 100BASE-T1: State of the Art of Communication Protocols and Experimental Evaluation of Time Synchronization using gPTP

Candidates:
Fabio Delbosco (S322244)

Referee:
Prof. Stefano Di Carlo
Prof. Alessandro Savino

Academic Year 2024 - 2025

A Laura che ha sempre creduto in me,
alla mia famiglia che mi ha sempre spronato
e spinto a dare il meglio
a chi mi vuole bene.

Fabio

Contents

List of Figures

List of Tables

Abstract

The increasing complexity of modern vehicles, driven by the integration of ADAS, infotainment, and connected services, demands communication networks with higher bandwidth and precise time synchronization capabilities compared to traditional CAN-based systems. In this context, Automotive Ethernet, and particularly the 100BASE-T1 (BroadR-Reach) standard, enables full-duplex communication over a single twisted pair, combining high data throughput with reduced wiring complexity. This thesis investigates the physical layer and the main Automotive Ethernet communication protocols — including TSN, SOME/IP, DoIP, and gPTP (IEEE 802.1AS) — focusing on their role in time-sensitive and service-oriented automotive architectures. An experimental setup was developed using two NXP S32K344-based boards, configured to communicate over a 100BASE-T1 point-to-point link. The gPTP synchronization stack provided by NXP was adapted to a bare-metal environment without AUTOSAR dependencies, enabling both hardware and software timestamping for network synchronization analysis. The work provides a complete experimental characterization of the implemented system, offering insights into the accuracy, stability, and practical limitations of gPTP synchronization in Automotive Ethernet networks and its suitability for future zonal architectures.

CHAPTER 1

Introduction

The continuous evolution of modern vehicles has led to a dramatic increase in the amount of data exchanged inside the in-vehicle network. Advanced driving assistance systems (ADAS), high-resolution cameras, radar and lidar sensors, connectivity services, and infotainment features all require reliable, high-bandwidth communication between Electronic Control Units (ECUs). As the level of autonomy and intelligence in vehicles increases, the in-vehicle communication infrastructure becomes a key enabler of safety, performance, and user experience.

Traditional automotive networks such as CAN, LIN, and FlexRay are no longer sufficient to handle these bandwidth demands. For this reason, Ethernet technology has been progressively introduced as the backbone of the in-vehicle communication system. Among the different variants standardized for automotive environments, **100BASE-T1** (IEEE 802.3bw) represents a crucial milestone, enabling 100 Mbit/s full-duplex communication over a single unshielded twisted pair (UTP) cable. This reduces weight, cost, and harness complexity while maintaining excellent signal integrity and EMC performance [12].

1.0.1 Safety-Critical Communication

Many new functions in the vehicle fall under the category of mission- and safety-critical systems:

- Powertrain and chassis control (braking, steering assistance)
- Perception and decision modules for autonomous driving
- Low-latency sensor fusion and actuation loops

These applications impose strict timing requirements: bounded latency, very low jitter, and guaranteed availability. Ethernet alone is not sufficient to ensure deterministic behavior, therefore additional protocols and mechanisms are required, such as:

- Time-Sensitive Networking (TSN)
- IEEE 802.1AS / gPTP time synchronization
- Service-oriented middleware such as SOME/IP

1.0.2 Architectural Evolution

To integrate both legacy fieldbuses and Gigabit-class Ethernet, the in-vehicle network architecture is undergoing a fundamental transformation.

Domain Architecture

Historically, ECUs have been grouped into functional domains: body, powertrain, infotainment, etc. Each domain typically has a gateway to route traffic to other parts of the vehicle. While effective for earlier vehicles, cross-domain communication creates bottlenecks as data volumes increase.

Zonal Architecture

The emerging *zonal architecture* restructures the wiring into geographical zones, each controlled by a zonal ECU. High-performance computing units act as a centralized processing cluster, while communication between zones is handled through a high-speed Ethernet backbone.

This approach significantly reduces harness length and weight, improves scalability, and aligns with future over-the-air (OTA) update and cloud connectivity strategies.

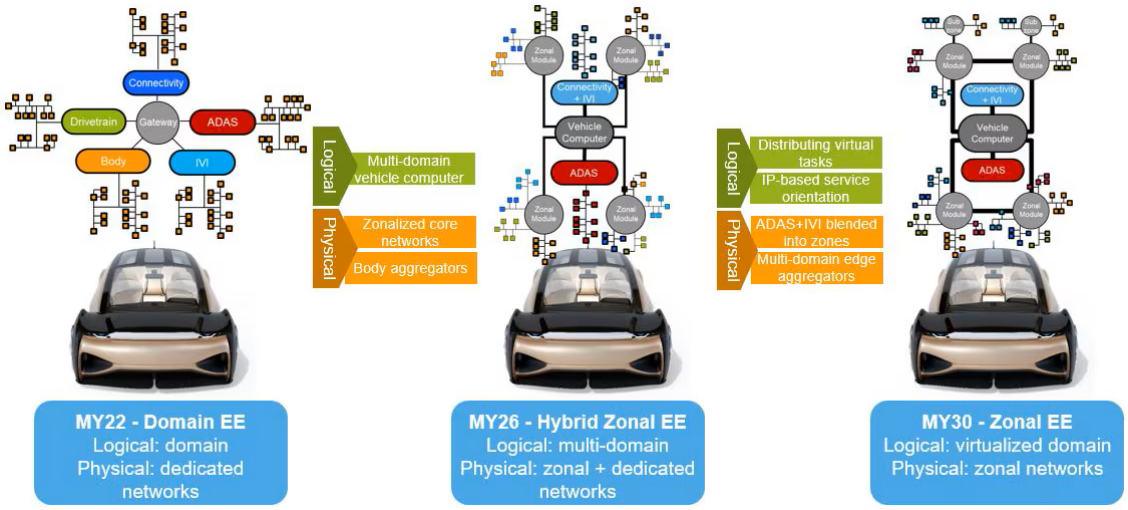

Figure 1.1: Different types of automotive architectures now in use [1].

1.0.3 Bandwidth Requirements

The exponential growth in network bandwidth is mostly driven by ADAS sensors and infotainment systems. Data production from a single perception sensor can exceed the capacity of entire legacy vehicle networks. Typical bandwidth demand includes:

- Radar: 100 Mbit/s
- Cameras: 500 Mbit/s per stream or higher
- Lidar: several Gbit/s

In this context, **100BASE-T1 is widely adopted for sensor connectivity and ECU backbone links** thanks to its bandwidth, robustness, and cost-effectiveness. It also serves as a fundamental building block toward higher-speed variants such as 1000BASE-T1 and Multi-Gig automotive Ethernet.
In summary, the shift toward software-defined vehicles and autonomous driving strongly depends on automotive Ethernet technologies. The 100BASE-T1 standard, combined with

TSN and gPTP, enables real-time communication and synchronized operations that are essential for safety-critical automotive applications. This thesis focuses on the behavior of a gPTP slave in a 100BASE-T1 network and the impact of packet loss and master unavailability on synchronization stability.

CHAPTER 2

Ethernet Automotive

The first Ethernet LAN was projected in at least 30 years ago by Robert Metcalfe at Xerox and the first published standards from the Digital Equipment Corporation of Intel in collaboration with Xerox.[13] In 1985 the Institute of Electrical and Electronics Engineers (IEEE) published the LAN standard with the initial number 802. Now the actual Ethernet standard is the 802.3 but, to assure conformance with the OSI model, some modification was done to the original Ethernet standard.

2.1 Network Topologies

Traditional in-vehicle networks, such as CAN, LIN historically relied on BUS topology that offers low cost and simplicity but limited bandwidth. Automotive Ethernet transition also introduce the need to redesign the network topologies as mentioned in the previous chapter. The following section describes briefly the most used Ethernet network topology also used in conventional IT networks, yet optimized for automotive constraint such as electromagnetic compatibility, deterministic latencies and cable weight:

- **Point-to-Point**: the foundation of the Ethernet standard, topology that works on the functionality of sender and receiver. It provides good quality connection and high bandwidth, such as the only destination of the message from the sender side is his receiver counterpart.

- **Mesh** : Every device included in the Mesh topology has a dedicated link to any another with a special connection on a dedicated channel, known as Link. Supposing to have N node, the total number of port in a node is N-1, therefore the total number of links in the network is given by N*(N-1)/2. The advantages are multiple: very fast communication; a robust architecture that can still manages to deliver the message to a node, even if the point-to-point connection with his neighbor is lost; fault detection is really fast and secure.
  Disadvantages arises in the high cable demand required to make all the connection between the nodes, also the maintenance of the network is difficult and high time consuming, redundancy reducing is difficult in such a redundant configuration.

- **Star**: Star topology comprehend a set of N nodes and a central HUB on which every node is connected to. Central HUB can be passive and forward passively all the message without a logic inside or active where all ports have transceivers to receive and send frame. The adoption of this topology is very large because the number of cables needed are equal to the nodes number, each node has only to have one dedicated port and also because of the fault identification and isolation is very

easy. Obviously the disadvantage are due to the centrality of the hub, in fact if it fails all the nodes became unable to communicate with all the other

- **Ring**: Ring topology forms a ring in which all node in the network has exactly 2 neighbor device. The communication arbitration works with the usage of Token Ring that is passed through the network, at any time the node that holds the token has the permission to send data in the network. Repeaters are used along the topology in order to avoid data loss in long message trip. This architecture has low possibility of collision, due to the token ring, high speed transmission and it has a low cost of installation. Of course this is not a robust topology as if one or more node fails entire chunk of the ring can be unreachable.
- **Tree**: takes some characteristics from star and bus topology. It presents with a gerarchical structures with lower nodes connected to HUBs as in the star topology and higher levels HUBs are connected each other with a sort of backbone that has as a parent HUB. The data flowing is bottom up from lower to higher levels. The advantages from the star topology is that the distances of a node from the central hub is decreased as the message travelled through devices instead of the cable. Maintenance and device adding in the network is easy. The disadvantages are also inherited from the Star topology as if the central bug fail the entire network fails and the cost is high due to the cable usage and difficult reconfiguration due to a device adding operation on the network.
- **Hybrid**: it is a combination of all the above descripted topology. This topology is extremely flexible as can be designed on the actual composition and problem of the network, it can be expanded easily adding new devices or entire new section. On the other hand HUBs in this configurations are very expensive as they had to support various type of communication protocols.

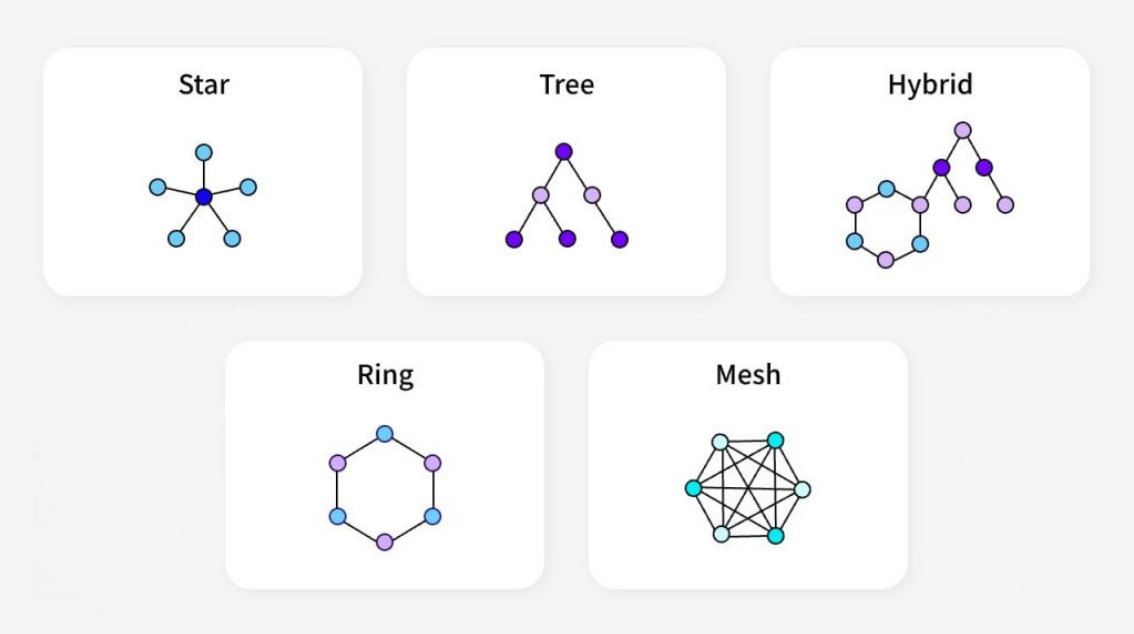

Figure 2.1: Types of network topologies [2].

2.2 BroadR-Reach and IEEE 802.3

IEEE 802.3 is the standard that defines Ethernet at the physical layer and the Logical Link Control (LLC) sublayer since its first edition in the 1980s. Over the decades, the 802.3 family has been continuously expanded with several physical layer variants, such as 100BASE-TX and 1000BASE-T, in order to meet the increasing demands for higher data rates and different transmission media. Modifications and updates are published as amendments and later integrated into consolidated editions of the standard, such as IEEE 802.3-2012, IEEE 802.3-2018, and the latest IEEE 802.3-2022.

The growing demand for bandwidth in the automation and automotive fields identified Ethernet as a suitable base technology, offering lighter and more robust cabling solutions and lower implementation costs. At the end of the 1990s, Broadcom developed a new transceiver solution enabling full-duplex communication over a single balanced twisted pair of conductors. This technology was named *BroadR-Reach*. It provided a 100 Mbit/s bandwidth over a single unshielded twisted pair (UTP) cable, using a reduced signal bandwidth and advanced techniques to minimize electromagnetic interference (EMI), making it ideal for automotive environments. In 2011, the OPEN Alliance published the BroadR-Reach specification, making it openly available to the industry in order to promote its integration and adoption in automotive applications [4].

Figure 2.2: Broadcom Inc. is a major technology company specializing in the telecommunications field [2].

In order to further standardize this new technology and ensure interoperability with traditional Ethernet switches, it became necessary to integrate BroadR-Reach functionality and specifications into the IEEE 802.3 framework. The IEEE 802.3bw Working Group [14] developed and evaluated the solution for transmitting a 100 Mbit/s data stream over a single twisted pair cable. The work concluded with the publication of *IEEE Std 802.3bw-2015*, which defines **100BASE-T1** as a new standard for industrial and automotive applications. After the publication of this amendment, the modifications related to 100BASE-T1 were incorporated into later consolidated versions of IEEE 802.3 (starting from IEEE 802.3-2018). These steps are typical of the IEEE standardization process:

- Initial development and publication as a technical amendment.
- Subsequent integration into consolidated versions of the standard.

The publication of IEEE Std 802.3bw-2015 marked a turning point in the evolution of Ethernet technology, introducing for the first time an officially recognized single-pair physical layer within the IEEE 802.3 family. The 100BASE-T1 standard inherited the

fundamental Ethernet architecture — same frame structure, MAC functions, and upper-layer compatibility — while redefining the physical transmission medium and signaling methods to fit the constraints of automotive environments.

In traditional Ethernet implementations such as 100BASE-TX or 1000BASE-T, data transmission is carried out over multiple twisted pairs using separate wires for each direction of communication. In contrast, 100BASE-T1 achieves full-duplex communication over a single balanced twisted pair by means of sophisticated digital signal processing (DSP) techniques.

This approach drastically reduces cable weight and connector complexity while maintaining deterministic performance and high electromagnetic compatibility (EMC), both essential for in-vehicle networks.

The IEEE 802.3bw amendment specifies the complete set of physical layer parameters: modulation scheme, link segment characteristics, and transceiver requirements. These specifications were defined with a clear focus on interoperability between different vendors, long-term reliability under automotive conditions (vibration, and noise), and compliance with the general IEEE 802.3 MAC service interface.

Figure 2.3 conceptually places 100BASE-T1 within the ISO/OSI model. The technology corresponds entirely to the **Physical Layer (Layer 1)**, while maintaining the same Logical Link Control (LLC) and Media Access Control (MAC) layers as other Ethernet variants.

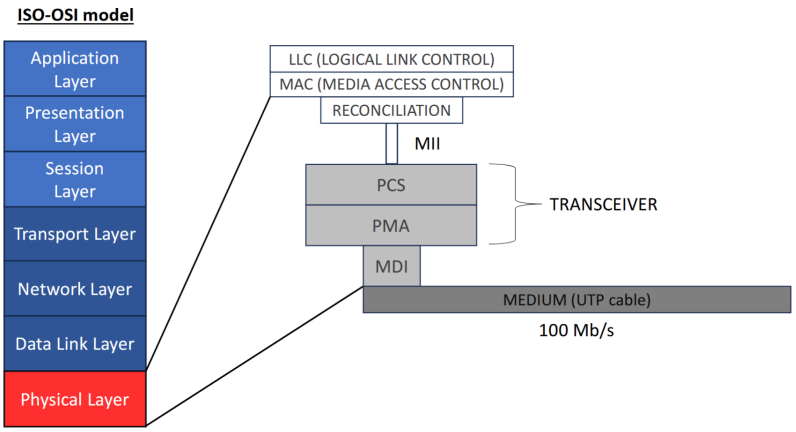

Figure 2.3: Placement of 100BASE-T1 within the ISO/OSI communication model.

The definition of the 100BASE-T1 physical layer also paved the way for subsequent developments such as 1000BASE-T1 (IEEE 802.3bp) and Multi-Gigabit Single Pair Ethernet (IEEE 802.3ch), which further extended the performance of single-pair Ethernet to higher data rates. Therefore, understanding the 100BASE-T1 physical layer provides the fundamental basis for comprehending the entire family of automotive and industrial Single Pair Ethernet (SPE) technologies.

The following section will therefore analyze the **Layer 1 structure** of 100BASE-T1, including the transceiver architecture, modulation schemes, and the synchronization mechanisms.

2.3 Physical Layer Overview

The **100BASE-T1** PHY operates in **full-duplex mode** over a single **unshielded twisted-pair (UTP)** cable, using **echo cancellation** to separate transmitted and received signals. It interfaces with the **MII**, **PCS**, and **PMA** sublayers to exchange data with the **MDI**,

which connects to the physical medium. The 100BASE-T1 leverages techniques derived from 1000BASE-T and 100BASE-TX PHYs, operating at a nominal rate of **100 Mb/s**, but introduces specific implementations of the PCS and PMA optimized for single-pair transmission.

- **MDI (Medium Dependent Interface)** — the first interface of the transceiver. It receives the electrical signal from the cable and delivers a corresponding digital stream to the upper modules.
- **PMA (Physical Medium Attachment)** — transmits and receives data between the PCS and the MDI, supporting link management and control functions.
- **PCS (Physical Coding Sublayer)** — manages data encoding and decoding between the MII and PMA, ensuring proper mapping and synchronization of digital symbols.
- **MII (Media Independent Interface)** — provides a standardized logical and electrical connection between the MAC and PHY layers.

2.3.1 Physical Coding Sublayer (PCS)

The **PCS** provides the logical mapping between the digital data of the MAC and the analog signal used on the medium. This module defines how the data are *scrambled* to maintain synchronization and minimize electromagnetic emissions, how bits are *encoded*, and how they are grouped into *symbols*.

Typical PCS operations include:

- Data encoding and scrambling to minimize electromagnetic emissions and ensure sufficient signal transitions for synchronization;
- Symbol management and error detection on the received data symbols;
- Frame delimiter and start-of-frame control for reliable transmission.

This sublayer is not identical across all IEEE 802.3 physical standards. For example, the **1000BASE-T** PCS uses a 5-level PAM encoding, while **100BASE-T1** employs a **PAM-3** modulation optimized for single-pair full-duplex transmission over unshielded twisted-pair cables.

By separating logical encoding from the physical signal interface, the PCS ensures design flexibility and supports a wide range of PHY implementations using a common MAC interface.

2.3.2 Physical Medium Attachment (PMA)

The **PMA** acts as the bridge between the symbol-level processing performed by the PCS and the electrical interface of the MDI. Its main functions are **serialization and deserialization (SerDes)**, **clock recovery**, and **link training**, which adapt the transceiver parameters to the characteristics of the physical channel.

During transmission, the PMA serializes the parallel data from the PCS and transmits a continuous bit stream synchronized with the transmission clock. During reception, it recovers the incoming signal, performs clock and data recovery, aligns the symbols, and reconstructs the bit stream to be passed to the PCS.

In modern Ethernet PHYs, the PMA also performs **echo cancellation** and **equalization** to mitigate signal attenuation, crosstalk, and impedance mismatch. These functions are crucial in full-duplex operation, where transmission and reception occur simultaneously

on the same wire pair. Effective noise reduction is particularly important in *automotive environments*, where cables and transceivers are exposed to strong electromagnetic interference and mechanical stress.

2.3.3 Media Independent Interface (MII)

The **Media Independent Interface (MII)** provides a standardized connection between the MAC and PHY. As its name suggests, it allows Ethernet controllers to integrate with different physical layers without hardware redesign.

The MII defines the set of *signals and pins* required to exchange data, control, and clock information between the MAC and PHY, for both transmit and receive paths. The number of pins depends on the data rate and specific implementation; variants such as **RMII** (Reduced MII) or **RGMII** (Reduced Gigabit MII) are often used to reduce pin count and simplify PCB layout.

2.3.4 Medium Dependent Interface (MDI)

The **MDI** represents the physical connection between the transceiver and the transmission medium, such as copper twisted-pair, coaxial cable, or fiber optics. It defines the **electrical characteristics**, **connector types**, and **impedance** of the link.

In the **100BASE-T1** standard, the MDI consists of a single unshielded twisted pair operating in full-duplex mode. Proper design of the MDI — including cable impedance control and connector layout — is crucial to meet the **EMC** and **signal integrity** requirements of automotive Ethernet systems.

2.4 100BASE-T1 Physical Layer Implementation

2.4.1 Full and Half-Duplex Communication

Ethernet communication can operate according to two main transmission modes: **half-duplex** and **full-duplex**. These two configurations differ in the way transmission and reception occur along the same communication medium.

Half-Duplex

In **half-duplex** mode, the communication channel is shared between transmission and reception, but only one direction can be active at a time. This means that a device must wait for the other to complete its transmission before being able to send its own data. A typical example of half-duplex communication is the operation of early Ethernet networks based on coaxial cable or hubs, where collision domains were managed by the *Carrier Sense Multiple Access with Collision Detection (CSMA/CD)* mechanism.

In these systems, if two devices attempt to transmit simultaneously, their signals collide, causing data corruption. Both transmitters must then stop and retry after a random delay. This approach, while simple and efficient for low traffic levels, limits the achievable bandwidth and increases latency, especially as the number of network nodes grows.

Full-Duplex

The evolution of Ethernet standards and the transition to switch-based topologies have made possible the introduction of **full-duplex** communication. In a full-duplex link, the two ends of the connection can transmit and receive simultaneously on the same physical medium. This effectively doubles the potential throughput compared to a half-duplex link of the same nominal data rate and completely eliminates collisions.

In full-duplex mode, each node can send frames independently without waiting for the line to be free, and there is no need for CSMA/CD mechanisms. However, simultaneous transmission and reception on the same cable introduces new physical challenges: the transmitted signal of a node inevitably couples into its own receiver path, generating what is known as **echo**. To correctly recover the information sent by the remote node, the receiver must therefore be capable of identifying and subtracting its own transmitted component from the incoming signal. This process is known as **echo cancellation** and is implemented in the *Physical Medium Attachment (PMA)* sublayer.

In 100BASE-T1 Ethernet, which operates over a single unshielded twisted pair (UTP) cable, full-duplex operation is achieved through careful analog design and sophisticated digital signal processing. Both ends of the link transmit continuously, with their signals superimposed in the medium. The received waveform is therefore a combination of the locally transmitted signal and the signal arriving from the remote end, each affected by the channel's propagation delay and attenuation.

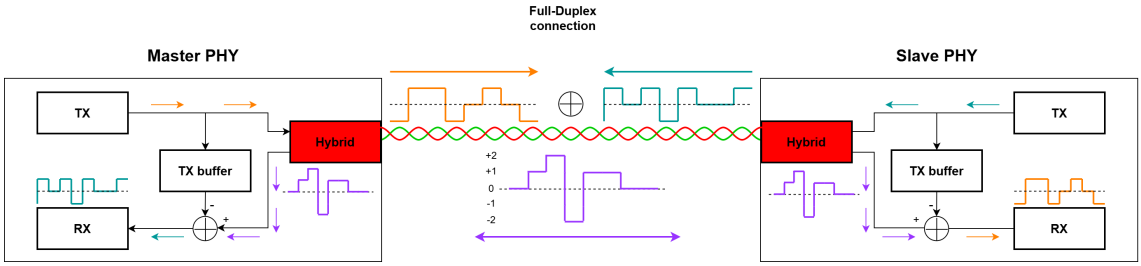

Figure 2.4: Architectural and functioning scheme of the Full-Duplex communication

From a theoretical point of view, let $s_A(t)$ and $s_B(t)$ be the signals transmitted respectively by nodes A and B, and $h(t)$ the channel impulse response. The signal received by node A can be expressed as:

$$r_A(t) = h(t) * s_B(t) + h_0(t) * s_A(t)$$

where $h_0(t)$ models the coupling of the transmitted signal into its own receiver (the echo path). An accurate estimate of $h_0(t)$ allows the receiver to subtract the echo and isolate $s_B(t)$, thus recovering the data sent by the other end.

The main advantages of full-duplex transmission are:

- **Increased throughput**: simultaneous bidirectional transmission effectively doubles the link capacity.
- **No collisions**: each direction operates independently, removing the need for collision detection.
- **Lower latency**: data can be transmitted immediately without waiting for the channel to become idle.

Full-duplex communication is therefore essential in modern high-speed Ethernet links, especially in automotive and industrial environments, where deterministic latency and continuous data flow are mandatory. The combination of full-duplex operation and advanced echo cancellation techniques represents one of the fundamental enablers of 100BASE-T1 technology, allowing robust, low-cost communication over a single twisted pair cable.

2.4.2 PAM3 Signaling and Line Coding

The **Physical Coding Sublayer (PCS)** of 100BASE-T1 is responsible for encoding and symbol generation, converting the data stream received from the **Media Independent Interface (MII)** into ternary symbols to be transmitted by the **Physical Medium Attachment (PMA)**.

At the interface with the MII, the PCS operates on a 4-bit data bus clocked at 25 MHz, corresponding to a raw throughput of 100 Mbit/s. However, before transmission, a **4B/3B** encoding process is applied: every 4 input bits are mapped into 3 output bits. This conversion introduces a rate mismatch, which is compensated by increasing the symbol rate at the PCS–PMA interface. To preserve the overall 100 Mbit/s bandwidth, the data frequency rises from 25 MHz to approximately 33.3 MHz.

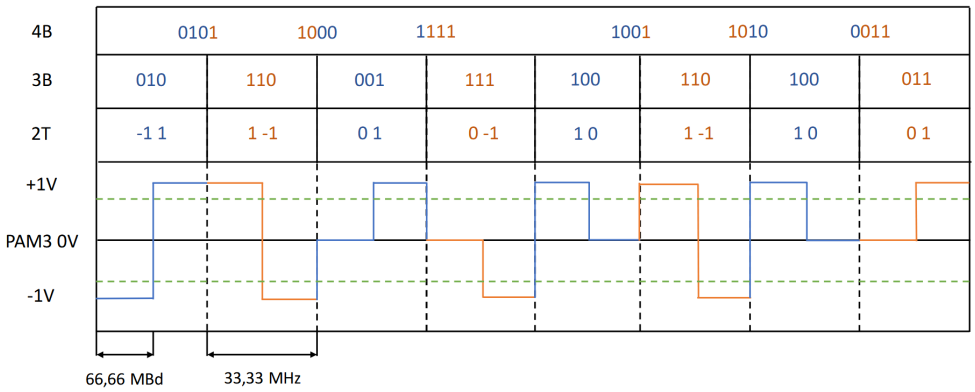

Figure 2.5: PAM-3 encoding and ternary symbol representation.

The main signals exchanged between the MII and the PCS are:

- **tx_enable_mii**: enables data transmission through the MII interface.
- **tx_error_mii**: indicates a symbol encoding error detected in the PCS.
- **TXD⟨3:0⟩**: 4-bit data nibbles exchanged between MII and PCS.
- **TX_CLK**: 25 MHz clock that synchronizes MII data transfers.

Between the PCS and PMA, data are exchanged through a 3-bit interface, synchronized at 33.3 MHz:

- **tx_data⟨2:0⟩**: data triplet bus from PCS to PMA.
- **tx_enable**: indicates valid data on the `tx_data` lines.
- **pcs_txclk**: 33.3 MHz transmission clock.

At the PMA output, the encoded symbols are mapped into voltage levels according to the **Pulse Amplitude Modulation with 3 levels (PAM-3)** scheme. Each ternary symbol can assume one of three voltage states, typically $[-1, 0, +1]$ V, transmitted as a differential signal across the twisted pair. A symbol transition thus represents one of three possible voltage changes, which encodes 3 bits per symbol pair.

Different transmission modes — **SEND_N**, **SEND_I** and **SEND_Z** — are used to represent the ternary symbol transitions, depending on the logical state of the transceiver during transmission and idle phases.

The transmission clock defines a **symbol pair period** of 30 ns, corresponding to a symbol rate of approximately 33.3 Mbaud. Given that each symbol pair encodes 3 bits, the resulting data rate is:

$$T_s = 30\,\text{ns} \qquad \Rightarrow \qquad f_s = \frac{1}{T_s} = 33.3\,\text{Msymbol/s}$$

$$R_b = f_s \times 3\,\text{bit/symbol} = 33.3\,\text{Msymbol/s} \times 3 = 100\,\text{Mbit/s}$$

During frame transmission, the PCS converts the **TXD⟨3:0⟩** nibbles into 3-bit symbols. Because Ethernet frame lengths are not always multiples of three, a **bit stuffing** mechanism is employed to complete the last symbol group. These extra bits are appended to the frame and discarded at the receiver side once the end-of-frame boundary is reached.

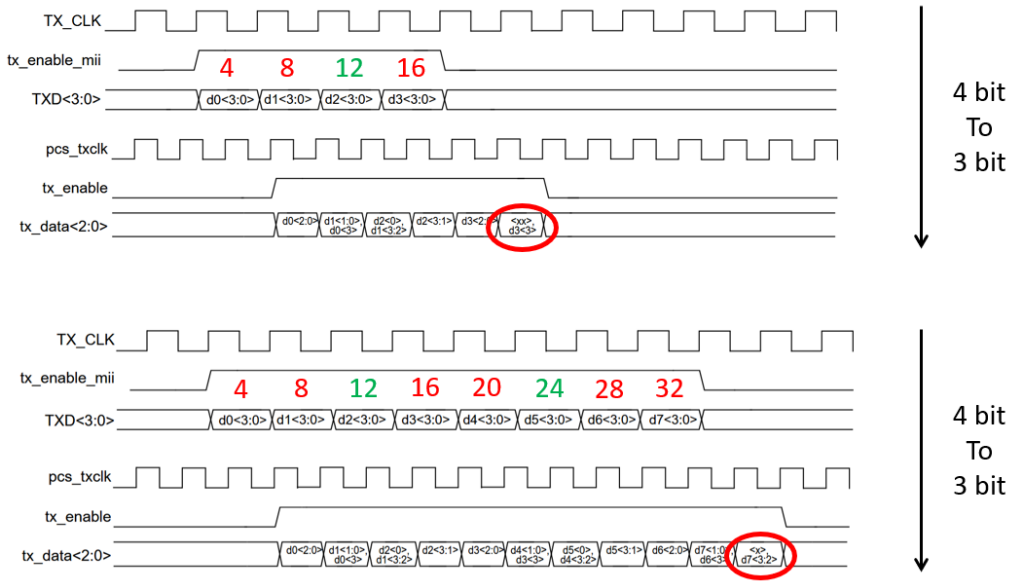

Figure 2.6: Example of 4B/3B conversion with bit stuffing to align frame boundaries. Source: [3]

The **12-byte Inter Packet Gap (IPG)** defined by the Ethernet standard ensures that the receiver can flush any residual stuffed bits from the PCS buffer before the next frame begins, avoiding FIFO overflow and maintaining symbol alignment.

In summary, the PAM-3 encoding and 4B/3B mapping provide an efficient compromise between spectral efficiency, EMI reduction, and hardware complexity. This scheme allows 100BASE-T1 to maintain a 100 Mbit/s throughput using a single unshielded twisted pair while keeping signal bandwidth within 66 MHz — a key factor for reliable automotive transmission channels.

2.4.3 Data Scrambling

Scrambling is employed as an additional processing step to enhance synchronization and maintain DC balance in the transmitted signal.

Scrambler

Build on a randomizing algorithm it is use to eliminate long strings of consecutive identical symbols transmitted on the medium. This technique allow to avoid the presence of spectral

lines, such as high peak, in the signal spectrum without changing the signal frequency rate. There are two main types of scrambler:

- **Self-synchronous** where the next scrambling state is dependent only on the prior n bits of the last output. It's counter part, **descrambler**, can aquire the correct scrambling state from the incoming signal.
- **Side-stream** a additional device is present to calculates the next state, depending only on the previous scrambling state and not also on the produced output. Descrambler chose a state either searching for a state that decodes a know pattern or by agreement at a known state with the scrambler.

Side-stream scrambler is adopted in the 100BASE-T1 protocol, it is implemented by a 33-bit LFSR able to generate pseudo random sequence that has same probability to have 0 or 1 in each position. This sequence will be then XORED with the data coming from the 4B/3B data converter.

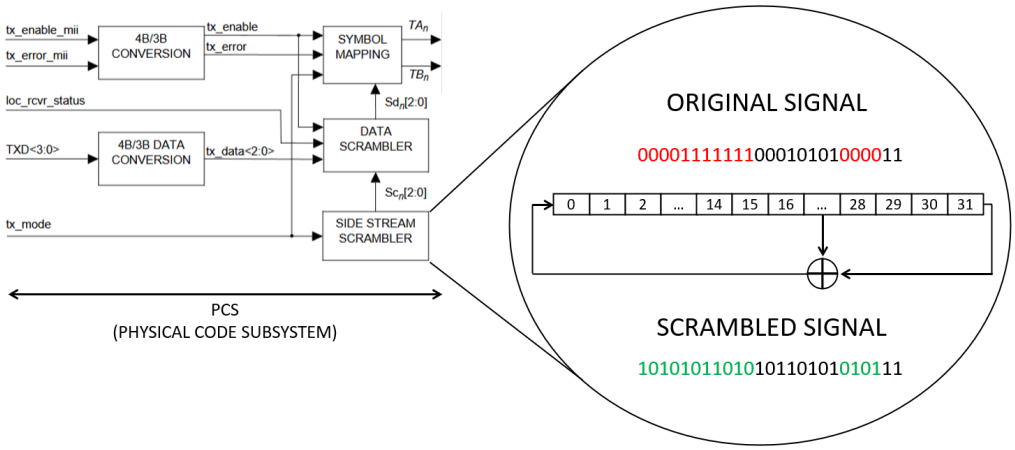

Figure 2.7: PCS logical design and scrambler functioning example. Source: [3]

Different polynomials are employed depening on the node role, depending on the value in the PMA_CONFIG.indication message. Implementation of master and slave PHY side-scramblers is formed by a shift register where bit are store in a register delay line. At time n is $SRC_n\,[32:0]$ and at each symbol time shift register is advanced by one position and new value is presented at $SRC_n\,[0]$. For obvious reason this register cannot ever shall be initialized with all 0's.

$$Master \quad \Rightarrow \quad g_M(x) = 1 + x^{13} + x^{33}$$

$$Slave \quad \Rightarrow \quad g_M(x) = 1 + x^{20} + x^{33}$$

To ensure proper data recovery, the scrambler and descrambler must operate in a synchronized manner, sharing an identical shift-register state at the start of communication. This synchronization is typically established during the training or IDLE phase, when known data patterns are exchanged. By observing the received symbols and comparing them with the expected scrambled values, the receiver gradually reconstructs the internal state of the transmitter's scrambler. Once the descrambler output remains consistent over several symbol periods, synchronization is considered locked. Some implementations also include automatic polarity detection: the receiver assumes a polarity, verifies descrambler convergence, and, if necessary, retries with the opposite assumption. This mechanism guarantees reliable state acquisition without requiring explicit transmission of the scrambler state.

2.4.4 Node Synchronization

Master & Slave

Node synchronization as to be assured between all the point to point connection in order to exploit the full-duplex communication mode. For this reason MASTER-SLAVE assignment for each link end is necessary for establishing timing control on each transceiver. Indeed to assure a good quality connection, slave device must be able to retrieve the clock signal from master's data and thus maintain a good synchronization. PHY mode can be chosen either by hardware switch or by setting the proper value in the dedicated control registers, present in each transceiver. Master-slave couplings will trigger errors during the **training phase**, resulting in an incorrect training procedure that will prevent the establishment of a connection between the two points.

Clock synchronization

In the **100BASE-T1 Ethernet** standard, there is *no dedicated* clock line shared by the MASTER. Instead, both data and timing information are transmitted over a *single differential* pair. As a result, the **SLAVE** device must recover the clock directly from the received data stream.

The **periodicity of signal transitions** — both rising and falling edges — is ensured by the **scrambler** in the transmitter node. This scrambler introduces sufficient data transitions to maintain clock recovery, provided that both the **scrambler** and **descrambler** are *perfectly synchronized* to guarantee reliable encoding and decoding operations.

Clock synchronization is achieved by a **phase detector and phase shifter** circuit, which continuously compares the *incoming clock* phase with the local *oscillator* running at **33.3 MHz**. Any detected phase difference is used to *adjust the local clock* so that it remains I with the transmitter's timing, ensuring stable and synchronized data communication between MASTER and SLAVE.

Operational Modes

Three operational mode are available to specify the sequence of code-groups sent by the transmitter.

- **SEND_N**: continuously asserted when the transmission of sequences of code-groups representing MII data stream, control mode or idle mode is to take place.
- **SEND_I**: continuously asserted in case of transmission of sequences of code-groups representing the idle mode
- **SEND_Z**: continuously asserted in case of transmission of consecutive zero is needed.

Various pattern mode are distringuishabke for their symbols composition and entropy, for example in SEND_N idle pattern will have low entropy than the data pattern. Those three mode are used to identify the various steps of the training phase and error occurrences.

Sd_n [2 : 0]	Ta_n	TB_n
000	-1	0
001	**0**	**1**
010	–1	1
011	**0**	**1**
100	1	0
101	**0**	**–1**
110	1	–1
111	**0**	**–1**

(a) Symbol mapping in **SEND_I** mode.

Sd_n [2 : 0]	Ta_n	TB_n
000	–1	–1
001	–1	0
010	–1	1
011	0	–1
Used for SSD/ESD	0	0
100	0	1
101	1	–1
110	1	0
111	1	1

(b) Symbol mapping in **SEND_N** mode.

Table 2.1: Comparison between **SEND_I** and **SEND_N** symbol mappings.

SSD/EDS code-sequence

SSD and ESD are 3 code sequences used, by the PMA only in SEND_N mode, to specify the start and the end of a packet frame. SSD is a 3 symbols code preceding the transmission od a frame. ESD is a 3 symbol code that follows the end of the frame and the deassertion of the TX_EN signal identifying the end of a transmission. ESD with error is also attached to the end of the frame signaling the presence of an error occurred during the transmission.

$$\begin{aligned} SSD &\Rightarrow (00), (00), (00) \\ ESD &\Rightarrow (00), (00), (11) \\ \text{ESD with } tx\_error &\Rightarrow (00), (00), (-1 - 1) \end{aligned}$$

2.4.5 Training Phase

As 100BASE-T1 leverages a full-duplex communication protocol, an initialization and training phase must be performed between the **MASTER** and **SLAVE**. During the training phase, both devices send data patterns that allow to:

1. Train the echo canceller with the transmitter signal, in order to be able to filter it out when the signal is received at the receiver side.
2. Lock the scrambler synchronization (see Section 2.4.3).
3. Synchronize the SLAVE clock to the MASTER clock retrieved from the data signal.
4. Adjust the signal conditioning in order to tune filters according to the received signal, which is essential to further eliminate noise sources or fine-tune other signal parameters.

This phase should last a maximum of 200 ms. Two timers are used to keep this constraint under control.

Link Establishment

The training phase is always initiated by the MASTER device, which transmits a training sequence in `SEND_I` mode. This mode allows the remote SLAVE to receive a predictable pattern used to calibrate internal circuits — including the echo canceller, equalizer, and clock synchronization — as described in IEEE 802.3bw Clause 96 [3]. The overall process can be summarized as follows:

1. The MASTER begins transmission in `SEND_I` mode, while the SLAVE remains in `SEND_Z`. During this interval, the MASTER uses the known training signal to adapt its own echo canceller and ensure stable transmission [15].
2. The SLAVE receives the idle pattern, checks for correct symbol decoding, and then trains its echo canceller. It also recovers the MASTER's clock from the data stream and locks its own scrambler to match the MASTER's state.
3. Once the recovered clock remains stable for approximately 1.8 µs, the SLAVE begins transmitting its own idle pattern in `SEND_I` mode. This allows the MASTER to confirm that the SLAVE's receiver is correctly synchronized.
4. When both sides complete all synchronization procedures, the `LINK_STATUS` register is set to 1, indicating that `LINK_UP` has been successfully achieved. The MASTER can then begin data or idle transmission in `SEND_N` mode.

If the `loc_rcvr_status` flag does not reach the `OK` state within 200 ms, the training phase is considered failed. In this case, the SLAVE enters the `SLAVE_SILENT` mode, and its transmitter switches to `SEND_Z`. The MASTER detects the continuous stream of zeros, interprets it as a failed training attempt, and restarts the initialization procedure.

A link drop may occur when an invalid ternary pair is detected, when the local receiver can no longer recover the clock, or in case of a physical fault such as a short circuit in the connection. Depending on the PHY implementation, a high rate of errors — such as `SSD`, `ESD`, `transmit_err`, or `receiver_err` — may also trigger a retraining sequence to reestablish the link. These mechanisms are designed to ensure robust operation under the high-noise and EMI conditions typical of automotive environments [**?**, **?**].

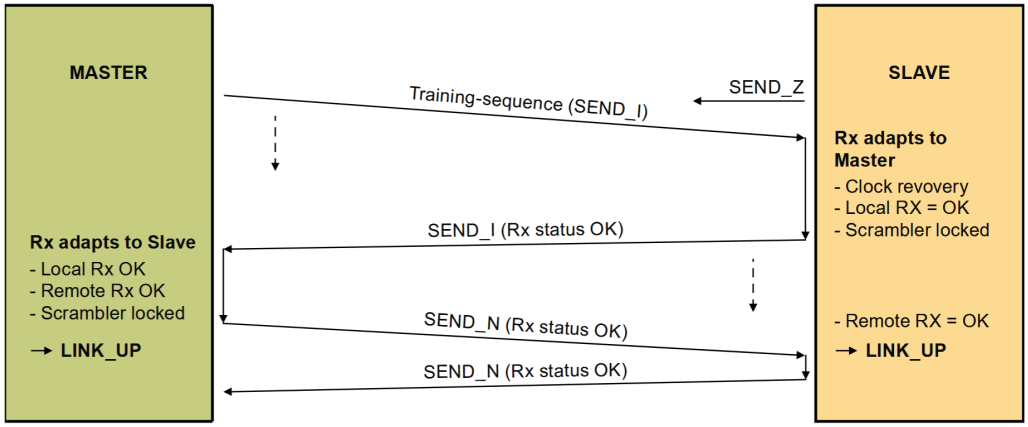

Figure 2.8: Link training and establishment procedure. Source: NXP datasheet, consistent with IEEE 802.3 specifications.

2.4.6 Cable Characteristics & connectors

The 100BASE-T1 standard specifies the use of a single unshielded twisted pair (UTP) cable with a **characteristic differential impedance** of 100 Ω ± 10%. This value is not arbitrary but results from both physical transmission constraints and the need for interoperability within the IEEE 802.3 Ethernet family.

The characteristic impedance of a transmission line depends on its distributed inductance L' and capacitance C' per unit length, and can be expressed as:

$$Z_0 = \sqrt{\frac{L'}{C'}}.$$

Maintaining the same impedance at the transmitter, cable, and receiver interfaces ensures proper **impedance matching**, preventing reflections and standing waves that would otherwise degrade the signal quality and cause inter-symbol interference (ISI). This matching condition is essential for reliable high-speed transmission, especially in **full-duplex operation** where simultaneous transmission and reception occur on the same medium.

Moreover, the use of a **balanced differential pair** with $Z_0 = 100\ \Omega$ guarantees a high level of **common-mode noise rejection**. In differential signaling, the two conductors carry equal and opposite signals, so any external electromagnetic interference (EMI) coupled into both wires is largely canceled at the receiver input. This property is fundamental in automotive environments, where cables are often routed near noisy sources such as power lines, motors, and switching converters.

However, despite its robustness, the 100BASE-T1 physical layer is limited to a maximum cable length of approximately **15 meters**. This restriction derives from the inherent sensitivity of full-duplex transmission over a single pair: as both ends transmit and receive simultaneously, **echo and near-end crosstalk** become dominant noise sources. Longer cable runs would exacerbate attenuation and phase distortion, making echo cancellation increasingly difficult and reducing the signal-to-noise ratio (SNR) below the required operational margin. Therefore, the 15 m limit represents a practical trade-off between link reliability, data integrity, and EMC performance within the intended automotive use case.

Finally, the choice of a 100 Ω impedance maintains compatibility with other twisted-pair Ethernet physical layers (e.g., 100BASE-TX, 1000BASE-T), simplifying transceiver design and allowing reuse of existing magnetics, connectors, and testing equipment optimized for this impedance range. Hence, the 100 Ω balanced twisted pair represents the optimal compromise between signal integrity, EMI immunity, mechanical simplicity, and cost efficiency for the 100BASE-T1 physical layer.

At the moment there are no standards that regulate the connector for the automotive ethernet protocol. Most used MDI in the automotive world are the H- MTD connector and the Mate-Net ones. There are several solutions for single and multiple male/female connector, all of them has to follow the automotive standards with a CPA connector and has to be waterproof. All compliant connectors have to follow the impedance of the cable and surely assure the datarate of the link connected to them.

MDI Connectors in Automotive Ethernet

At present, there are no IEEE standards that define a unique connector type for the **100BASE-T1** automotive Ethernet physical layer. Instead, the choice of the **Media Dependent Interface (MDI)** connector is left to manufacturers, provided that the interface ensures compliance with electrical and mechanical requirements defined by the standard and automotive-grade specifications.

Among the most widely used solutions in the automotive market are the **H-MTD** connector, developed by Rosenberger, and the **Mate-Net** connector, developed by TE Connectivity. Both solutions are designed for high-speed differential transmission over a 100 Ω balanced twisted pair and meet the environmental and mechanical constraints typical of automotive environments, including vibration resistance, humidity protection, and secure locking through **CPA (Connector Position Assurance)** mechanisms.

All compliant connectors must maintain impedance continuity with the transmission line to prevent signal reflections and ensure proper **signal integrity**. They must also

support the data rate required by the physical layer (up to 100 Mbit/s for 100BASE-T1) and, in many cases, are designed with a higher bandwidth margin to support potential reuse for faster protocols (up to several Gbit/s).

Table 2.2: Main characteristics of H-MTD and Mate-Net automotive Ethernet connectors.

Connector	**Main Characteristics**
H-MTD (Rosenberger)	• 100 Ω differential impedance • Frequency range up to 20 GHz • Data rates up to 56 Gb/s • Waterproof design • CPA locking system • Mating cycles ≥ 25
Mate-Net (TE Connectivity)	• Supports up to 6 Gb/s data rate • NanoMQS terminal system • Compact, sealed single-port design • Automotive-grade waterproofing

In summary, both connectors satisfy the stringent automotive EMC and environmental requirements while preserving the 100 Ω impedance continuity required by 100BASE-T1 links. The **H-MTD** connector provides higher frequency capabilities and scalability up to multi-gigabit links, making it more suitable for future Ethernet generations (e.g., 2.5GBASE-T1 or 10GBASE-T1). Conversely, the **Mate-Net** connector offers a more compact and cost-efficient solution, ideal for low-to-medium data rate applications such as sensors or control modules.

2.4.7 Echo Cancellation

In full-duplex communication, signals in both directions between two nodes propagate simultaneously over the same 100 Ω balanced cable. Even though the line ends are terminated with 100 Ω, signal reflections can still occur due to impedance mismatches or mechanical imperfections in the cable or MDI connector.

To mitigate this, the receiver employs advanced signal reconstruction and filtering techniques, most importantly **echo cancellation**. The echo canceller continuously adapts to the received signal to estimate and remove the transmitter's own reflected signal from the incoming data.

Echo is a key factor limiting the maximum cable length. The canceller must wait for the reflected signal to evaluate its contribution. For example, for a cable of length L [m], the round-trip propagation delay can be estimated as:

$$\text{Delay [ns]} = 2 \cdot (5 \text{ ns/m} \cdot L \text{ [m]}) \tag{2.1}$$

where the factor 5 ns/m corresponds to the propagation velocity in typical automotive twisted pair cables. For a 15 m cable, the resulting delay is 150 ns, which is also affected by **insertion loss** and other noise sources. If the cable is too long, the echo canceller cannot reliably estimate the reflected signal, increasing the risk of signal degradation and bit errors.

Typical values for automotive Ethernet links, taken from [5], are:

- **Insertion loss:** ~ -1.0 dB, representing the attenuation of signal amplitude along the cable.
- **Return loss:** ~ -20 dB, indicating the fraction of signal reflected back due to impedance discontinuities.

Figure 2.9 illustrates the concept, showing the transmitted signal (TX), the received signal (RX), and the echo component of the transmitter that must be cancelled to preserve signal integrity.

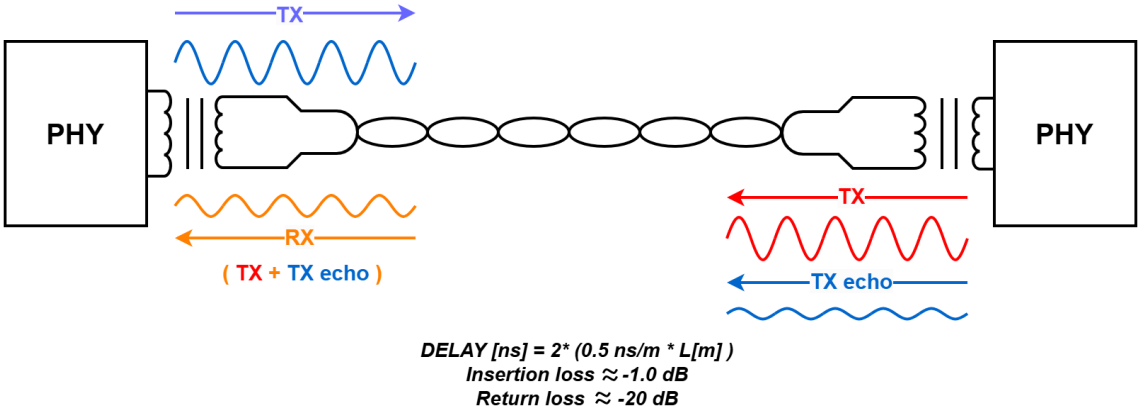

Figure 2.9: Illustration of transmitted (TX) and received (RX) signals including the transmitter's echo, highlighting the role of echo cancellation.

Data Path and Propagation Delay

The overall **data delay** in a 100BASE-T1 link is the sum of multiple contributions along the transmission path. Distinct delays are observed in the transmit and receive directions, as the received signal must pass through additional processing blocks such as the **echo canceller** and the **equalizer**, which introduce extra latency.

The **reception delay** (path from MDI to MII) typically ranges between 760 ns and 941 ns, depending on the implementation and adaptive filter complexity. Conversely, the **transmission delay** (path from MII to MDI) is lower, varying between 140 ns and 300 ns.

In addition to PHY-internal latency, there is also the **propagation delay** introduced by the physical cable, which can be approximated as 5 ns/m. For the maximum recommended cable length of 15 m, this corresponds to a delay of approximately 75 ns.

Therefore, the overall worst-case propagation time from one node to another can be expressed as:

$$T_{\text{propagation, TX}} = T_{\text{cable}} + T_{\text{PHY, TX path}} = 75\ \text{ns} + 300\ \text{ns} = 375\ \text{ns} \tag{2.2}$$

$$T_{\text{propagation, RX}} = T_{\text{cable}} + T_{\text{PHY, RX path}} = 75\ \text{ns} + 941\ \text{ns} = 1.016\ \mu\text{s} \tag{2.3}$$

$$T_{\text{total, worst-case}} = T_{\text{propagation, TX}} + T_{\text{propagation, RX}} = 1.391\ \mu\text{s} \tag{2.4}$$

These values highlight that the receive path is inherently slower due to the digital signal processing required for echo cancellation and equalization. Understanding these timing

characteristics is essential for evaluating system-level latency in time-sensitive automotive Ethernet networks.

2.4.8 DC Isolation

The **ESD protection module** is responsible for safeguarding the transceiver from possible **electrostatic discharges (ESD)** or transient voltages that may be induced through the connected cables. Proper ESD protection is essential to ensure compliance with automotive EMC standards and to guarantee long-term reliability of the physical layer.

Capacitors **C1** and **C2**, each with a value of 100 nF, provide **DC isolation** between the transceiver differential pins and the external line. These capacitors block any DC offset while allowing high-frequency data signals to pass, ensuring the correct AC coupling between the PHY and the cable.

Another key protection element in the interface is the **Common Mode Choke (CMC)**, which filters out **Electromagnetic Interference (EMI)** and suppresses common-mode noise that may affect the data lines. The CMC thereby improves the overall signal integrity and reduces emissions radiated by the link, a critical aspect in the automotive environment.

The resistors **R1** and **R2**, typically 50 Ω each, ensure a proper impedance matching by providing:

- **50 Ω single-ended termination** for each signal line.
- **100 Ω differential termination** for the differential pair.

Accurate termination and impedance matching are fundamental to minimize signal reflections, echo noise, attenuation, and ultimately to prevent data packet loss.

Figure 2.10 shows the typical DC isolation and protection network, as specified by the **OPEN Alliance IEEE 100BASE-T1 System Implementation Specification** [4].

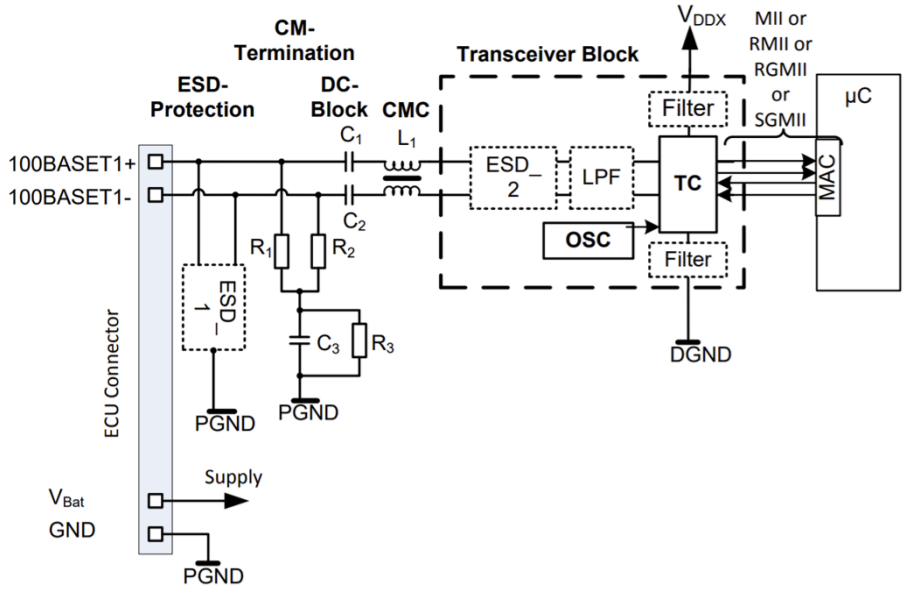

Figure 2.10: Typical DC isolation and protection circuit for 100BASE-T1, including ESD protection, AC coupling capacitors, common-mode choke, and differential termination resistors [4].

2.4.9 Fault Tolerances

According to the IEEE 802.3 standard, the **Media Dependent Interface (MDI)** is required to tolerate a variety of electrical stress conditions without sustaining permanent damage. In practice, this means that the twisted-pair lines must withstand short circuits either between the two conductors or between any conductor and ground or supply potentials (up to +50 V DC), with the source current limited to 150 mA, for an indefinite period of time. Once the fault condition is removed, the transceiver must be capable of resuming normal operation without performance degradation.

In addition to short-circuit tolerance, the MDI must also be resilient against high-voltage transient disturbances and **Electrostatic Discharges (ESD)** as defined by the automotive application requirements. These robustness constraints ensure the reliability of the physical layer in harsh environments typical of in-vehicle networks, where frequent transient events can occur due to switching loads, inductive components, or electrostatic coupling through the cable harness.

A further reliability feature of the 100BASE-T1 link is the automatic handling of **polarity inversion**. If the differential pair is connected with reversed polarity, the **slave PHY** automatically detects the condition and compensates it internally by swapping the polarity of its transmitter and receiver paths. This mechanism guarantees that no damage or loss of communication occurs due to incorrect wiring during installation or manufacturing.

Finally, if the link is lost or becomes unstable, the PHY initiates a new **training phase** to re-establish the connection. During this process, the transceivers exchange synchronization patterns to adapt their internal equalization and echo-cancellation filters, restoring a stable communication channel without manual intervention.

Media Independent Interfaces and PHY Integration

The automotive-Ethernet single-pair physical layer defined by IEEE 802.3 Clause 96 (100BASE-T1) mandates a well-defined interface between the MAC/domain controller and the PHY device. According to the standard, the PHY must support full-duplex communication over a single balanced twisted pair cable and integrate seamlessly with the MAC via a "Media Independent Interface" (MII) or one of its reduced variants. [3]

These specify, for example, that the MAC-PHY interface may be implemented in standard MII, Reduced MII (RMII) or even RGMII modes, as long as the interface meets timing, signalling and EMC constraints of the automotive environment.

Standard MII In its classical implementation, the MII interface provides a 4-bit parallel transmit bus (TX[3:0]) synchronized at 25 MHz, a 4-bit receive bus (RX[3:0]), an error signalling line and an enable/strobe line indicating the PHY's readiness to transmit data. This configuration supports 100 Mbit/s throughput when used in conjunction with a PHY compliant to Clause 96.

Reduced MII (RMII) To reduce pin-count and simplify PCB routing – a key objective in automotive ECUs – the RMII variant halves the number of data lines by multiplexing signals and uses a faster reference clock (typically 50 MHz). This allows the MAC-PHY connection to maintain the same nominal data rate while optimizing footprint and cost.

RGMII and other variants For higher-speed PHYs or more advanced embedded systems, the Reduced Gigabit MII (RGMII) is often supported. Although not strictly required

for 100BASE-T1, many PHYs intended for automotive networking include RGMII in order to support future scalability (for example 1000BASE-T1). Automotive PHYs provide "xMII flexibility" supporting MII, RMII, RGMII, SGMII and others. [5]

2.4.10 Basic node Setup Review

The basic architecture of a 100BASE-T1 node consists of a **Physical Layer (PHY)** device responsible for transmitting and receiving data over the differential pair, and a **Microcontroller Unit (MCU)** or **MAC controller** managing higher-level protocol functions. The communication between these two subsystems is implemented through a **Media Independent Interface (MII)**, which ensures a standardized data exchange between the digital MAC domain and the analog physical medium.

The standard MII defines a 4-bit parallel data bus (`TX[3:0]`) synchronized with a 25 MHz clock, complemented by control lines for transmission enable and error indication. A symmetric structure is used for reception through the `RX[3:0]` bus. This interface supports a theoretical data rate of 100 Mbit/s, corresponding to a 25 MHz data clock and 4-bit word width.

Inside the transceiver, the transmitted data undergoes a series of signal processing stages. The 4-bit nibbles are first converted into 3-bit symbols and serialized at an internal clock rate of approximately 33 MHz, preserving the overall throughput of 100 Mbit/s. The data stream is then **encoded using ternary signaling** and transmitted over the differential link with three voltage levels $\{-1, 0, +1\}$, resulting in a peak-to-peak amplitude below 2.2 V.

On the receiving side, the PHY performs **echo cancellation**, **equalization**, and **clock recovery** to reconstruct the transmitted signal while compensating for channel losses, impedance mismatches, and reflections along the twisted pair. These operations are essential to ensure signal integrity and compliance with the stringent electromagnetic and timing requirements of automotive environments.

A high-level representation of the complete physical layer architecture, from the MAC interface to the MDI and cable connection, is shown in Figure 2.11, adapted from the Texas Instruments 100BASE-T1 implementation guide [**?**].

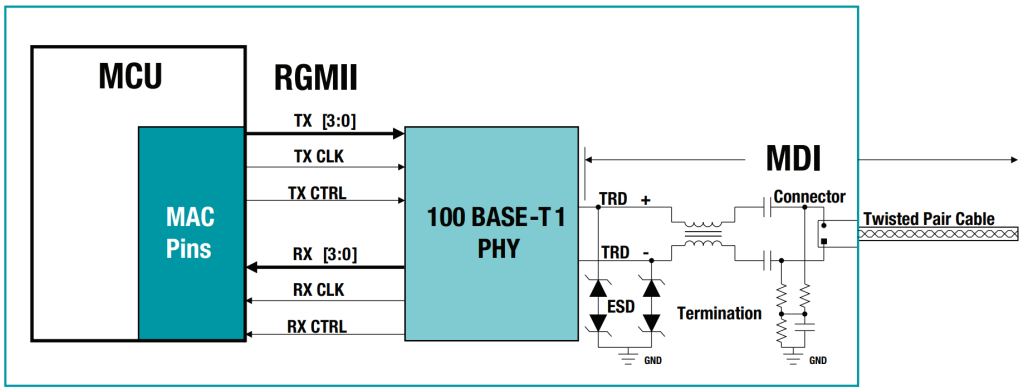

Figure 2.11: Overview of the 100BASE-T1 physical layer architecture, showing the interconnection between MAC, MII, PHY, MDI, and the differential cable link (adapted from [5]).

2.5 100BASE-T1 Logical Link Control Layer

The *Data Link Layer (DLL)* is responsible for the framing, transmission, and reception of data packets over the physical medium. It acts as an intermediary between the Network Layer and the Physical Layer, ensuring that data units are properly encapsulated and synchronized for reliable communication.

According to the IEEE 802.3 standard, the Data Link Layer is subdivided into two functional sublayers:

- **Media Access Control (MAC):** manages access to the shared physical medium and defines the rules for frame transmission and reception. It appends addressing information (MAC addresses) and error detection codes to the frame, coordinating the communication timing with the underlying physical interface.
- **Logical Link Control (LLC):** provides the interface between the Data Link Layer and the Network Layer. It handles frame delimitation, flow control, and logical addressing for upper-layer protocols. In automotive Ethernet implementations, the LLC typically offers minimal overhead, as the protocol is designed to maximize throughput and minimize latency, without performing retransmission or error recovery mechanisms.

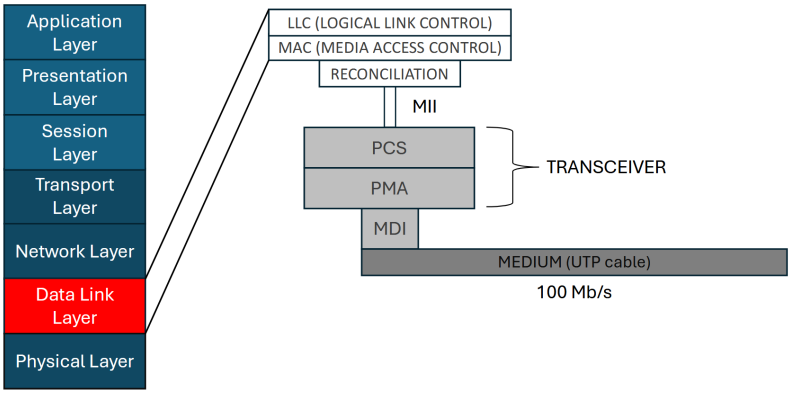

Figure 2.12: Placement of 100BASE-T1 within the ISO/OSI communication model.

In the following sections, the fundamental characteristics and functionalities of these two sublayers are described in detail, emphasizing their role in the implementation of the 100BASE-T1 communication protocol.

Communication primitives

The Ethernet concept of point-to-point communication means that the every station can only talk and send message to the master/slave end point connected to the other end of the link. The task of handling routing and adding destination addresses is entrusted to the upper Network IP layer. MAC has 2 primitives which can be used to communicate with the lower physical layer and signaling a reception of a message or request to send one:

- **MA_DATA.request**: This primitive defines the transfer of data from the MAC client entity to a single peer entity or multiple peer entities in the case of group addresses.

- **MA_DATA.indication**: : The primitive is passed from the MAC sublayer entity to the MAC client entity to indicate the arrival of a frame to the local MAC sublayer that is destined for the MAC client. Frame is reported only if they are validly formed, received without error, and their destination address designates the local MAC entity.

2.5.1 MII Data Stream and Frame Structure

The *Media Independent Interface (MII)* 2.4.9 plays a crucial role as it acts as the bridge between the MAC and the Physical Layer. The MII is designed to make the differences among various physical media transparent to the MAC sublayer, providing a standardized mechanism for the exchange of data between the two entities. The data stream format defines the architecture used to transmit and receive messages through the MII interface.

⟨**INTER-FRAME**⟩⟨**PREAMBLE**⟩⟨**SFD**⟩⟨**DATA**⟩⟨**EFD**⟩

- ⟨**INTER-FRAME**⟩: an idle period that provides an observation window during which no data activity occurs on the MII. It ensures adequate spacing between consecutive frame transmissions.
- ⟨**PREAMBLE**⟩: a bit sequence transmitted before the start of a frame. It consists of seven octets of alternating 1s and 0s, used to allow the receiver's clock to synchronize with the incoming data stream:

  10101010 10101010 10101010 10101010 10101010 10101010 10101010

- ⟨**SFD**⟩ **(Start Frame Delimiter)**: a unique bit pattern that follows the preamble and indicates the start of the actual frame data. It allows the receiver to align with the byte boundary:

  10101011

- ⟨**DATA**⟩: the payload field containing the actual frame data to be transmitted.
- ⟨**EFD**⟩ **(End Frame Delimiter)**: marks the end of the frame and consists of the deassertion of the control signal in the MII, indicating that the transmission is complete.

2.5.2 Ethernet Frame Composition

Throughout the evolution of Ethernet, additional capabilities have been introduced to support multiple Data Link Layer protocol encapsulations within the *MAC Client Data* field. As a result, several frame formats now coexist under the IEEE 802.3 standard, differing primarily in the interpretation of the *Length/Type* field and the presence of optional tags (e.g., VLAN or Q-tagging).

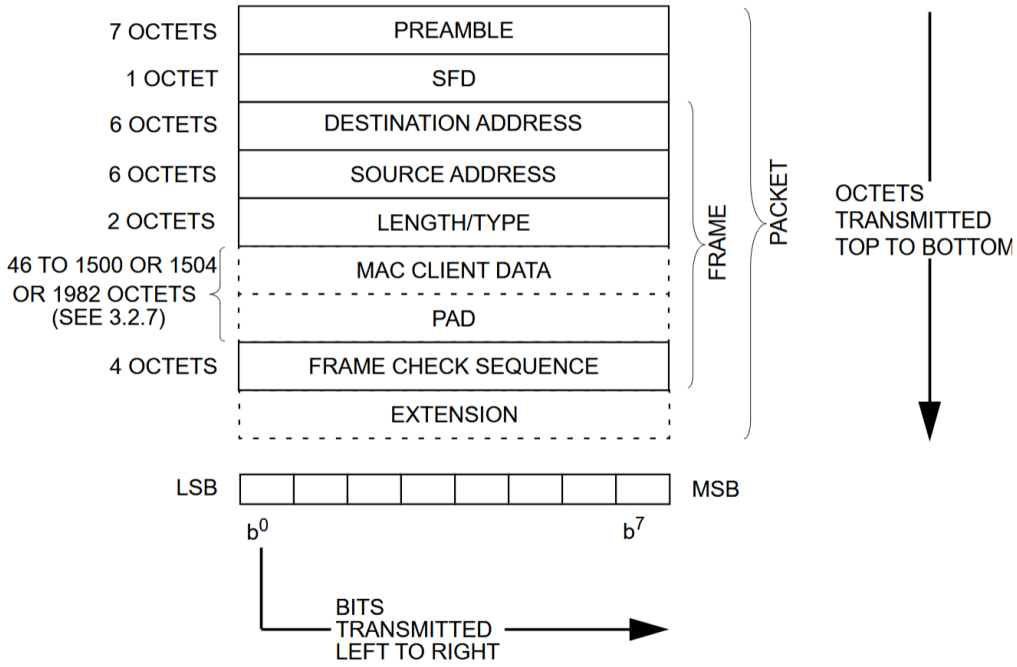

Figure 2.13: Ethernet frame architecture take from IEEE 802.3 standard [3]. Transmission is performed LSB first for each byte. Byte transmit order is instead MSB first.

The general structure of a standard Ethernet frame is shown below:

- **Preamble**: announces to the receiver the arrival of a new packet. It allows clock synchronization prior to the start of the actual frame.
- **Start Frame Delimiter (SFD)**: marks the beginning of the MAC frame and signals that the following bits correspond to frame content. These first two fields (*Preamble* and *SFD*) are inherited directly from the MII data stream format.
- **Destination Address**: identifies the intended recipient of the frame. It may represent a unicast, multicast, or broadcast address.
- **Source Address**: specifies the hardware address of the transmitting node.
- **Length/Type**: indicates either the length of the MAC Client Data field (for IEEE 802.3 frames) or the type of upper-layer protocol encapsulated (for Ethernet II frames, e.g., IPv4, IPv6, ARP).
- **MAC Client Data**: contains the payload of the frame, i.e., the actual data delivered by higher-layer protocols.
- **PAD**: optional padding bytes added when the payload is shorter than the minimum Ethernet frame size, ensuring compliance with the 64-byte minimum length requirement.
- **Frame Check Sequence (FCS)**: a 32-bit *Cyclic Redundancy Check (CRC)* value used for frame integrity verification and error detection at the receiver side.

Frames are transmitted with the **Most Significant Byte (MSB)** first. The payload field can range from **46 to 1500 bytes**, resulting in a total frame length of 64 to 1518 bytes (or up to 1522 bytes when VLAN tagging is used). At a transmission rate of **100 Mb/s**, a maximum-size Ethernet frame (12,336 bits) requires approximately **123.4 μs** to be fully transmitted over the medium.

Destination and Source Address

The *MAC address* is a low-level hardware identifier consisting of a 48-bit value uniquely associated with each device connected to an Ethernet network. It enables the identification of individual nodes for data transmission and reception at the Data Link Layer.

This address is divided into three main parts:

- **I/G (Individual/Group) bit**: the least significant bit of the first octet. It defines whether the address refers to a single device (*unicast*) or to a group of devices (*multicast*).
- **U/L (Universal/Local) bit**: the second least significant bit of the first octet. It specifies whether the address has been globally assigned by the IEEE (universal) or locally administered within the network.
- **Remaining 46 bits**: these are split between the *Organizationally Unique Identifier (OUI)* — a 24-bit code assigned by the IEEE to identify the manufacturer — and the *Network Interface Controller (NIC)-specific identifier*, which uniquely distinguishes each device produced by that manufacturer.

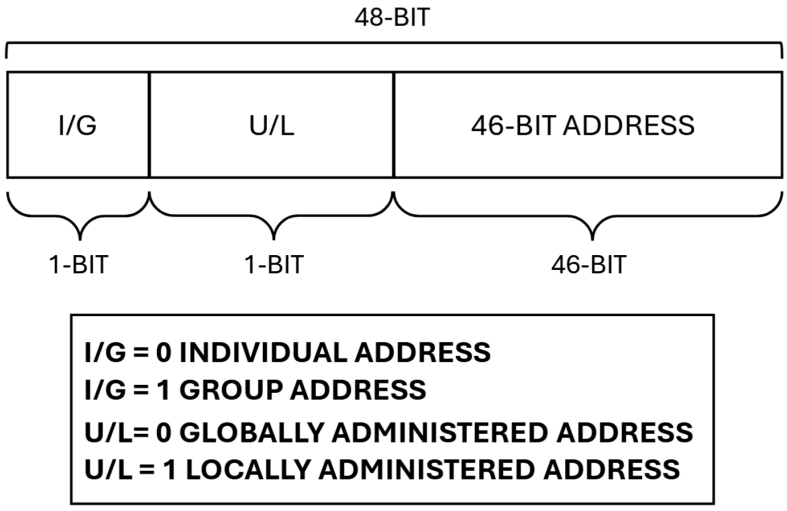

Figure 2.14: Structure and bit significance of the MAC address.

In an Ethernet frame, two MAC addresses are always specified:

- the **Destination Address**, identifying the recipient of the frame;
- the **Source Address**, indicating the node that originated the transmission.

A destination address composed entirely of binary ones (`FF:FF:FF:FF:FF:FF`) represents the **broadcast address**, meaning that the frame is intended for all devices on the network segment.

Unicast MAC Address

A **Unicast MAC address** identifies a single network interface within a local network. Frames sent to a unicast address are intended for one specific destination node only.

There are two types of unicast MAC addresses, depending on how they are assigned:

- **Universally Administered Address (UAA)**: The second least significant bit (LSB) of the first octet is set to `0` (`00:XX:XX:XX:XX:XX`).

  - It is globally unique and assigned by the IEEE to the device manufacturer.
  - The first 24 bits correspond to the *Organizationally Unique Identifier (OUI)*, which identifies the vendor or manufacturer.
  - The remaining 24 bits (in the case of MAC-48 and EUI-48) or 40 bits (for EUI-64) are assigned by the manufacturer to uniquely identify each interface.

- **Locally Administered Address (LAA)**: The second least significant bit (LSB) of the first octet is set to `1` (`02:XX:XX:XX:XX:XX`).
  - This address type is not globally unique and is typically assigned manually within a private or test network.
  - It is often used for virtual interfaces or reconfiguration purposes during development and testing.

Multicast and Broadcast MAC Addresses

Multicast addresses are specific MAC addresses that allow a network interface card (NIC) to receive frames addressed to a defined group of devices. They are commonly used for protocol-level communication, service discovery, or vendor-specific control messages.

When a switch receives an Ethernet frame with a multicast destination address, it forwards the frame to all nodes that belong to the corresponding multicast group. Multicast addresses are identified by having the least significant bit (LSB) of the first octet set to `1`.

Example: `01:XX:XX:XX:XX:XX`

Broadcast addressing is a special case of multicast in which all devices on the network are the intended recipients of the frame. A broadcast frame is identified by a destination MAC address in which all bits are set to one:

`FF:FF:FF:FF:FF:FF`

In this case, every NIC on the network receives and processes the encapsulated message.

Length and EtherType Field

The **Length/Type** field of an Ethernet frame consists of 2 octets (16 bits) and serves to indicate either the length of the MAC Client Data field or the EtherType of the encapsulated protocol. Its interpretation depends on the value contained in the field:

- If the value is less than or equal to 1,500 decimal (`05DC` hex), the field specifies the **length in bytes of the MAC Client Data** field that follows.
- If the value exceeds 1,536 decimal (`0600` hex), the field identifies the **EtherType** of the upper-layer protocol encapsulated in the frame.

The interpretation of the field is mutually exclusive: it indicates either the length of the data field or the EtherType of the frame.

When used as a length indicator, it is the responsibility of the MAC client to ensure that proper **padding** is applied if the data field is shorter than the protocol's minimum size. Regardless of the interpretation, if the MAC Client Data field is shorter than required, a **Pad field** — a sequence of octets — is added immediately before the **Frame Check Sequence (FCS)** to meet the minimum frame length requirement.

Common Automotive EtherTypes

Automotive Ethernet frames typically use well-known EtherTypes such as IPv4, IPv6, and ARP for general message packetization and forwarding. In addition, some EtherTypes are reserved for automotive-specific protocols, including infotainment, AVB/TSN, and emerging V2X (Vehicle-to-Everything) communications.

Table 2.3 summarizes the most common EtherTypes found in automotive Ethernet networks:

Table 2.3: Common EtherTypes in automotive Ethernet networks

Protocol / Usage	**EtherType (Hex)**	**Description**
IPv4	0x0800	Internet Protocol version 4
IPv6	0x86DD	Internet Protocol version 6
ARP	0x0806	Address Resolution Protocol
VLAN – Single Tag	0x8100	IEEE 802.1Q single VLAN tag
VLAN – Double Tag	0x9100	IEEE 802.1Q double VLAN tag (Q-in-Q)
Multiple VLAN Reservation Protocol	0x88F5	Used for reserving multiple VLAN streams
SOME/IP (ISO 17220)	0x22F0	Scalable service-Oriented Middleware over IP
AVB / Generalized Precision Time Protocol	0x88F7	Audio Video Bridging / TSN synchronization
Multiple Stream Reservation Protocol	0x22EA	AVB stream reservation protocol
Wave Short Message Protocol	0x88DC	V2X short message communication

V2X (Vehicle-to-Everything) enables vehicles to communicate with other environmental entities, including other vehicles, buildings, and infrastructure, typically via wireless communication channels. This set of EtherTypes allows automotive devices to correctly identify and process the payload of each frame, supporting both general networking functions and automotive-specific applications such as infotainment, TSN, and V2X messaging.

MAC Client Data Field

The **MAC Client Data field** contains a sequence of octets representing the actual payload to be transmitted from one node to another. Full data transparency is provided, meaning that any arbitrary sequence of octet values may appear in this field, up to a predefined maximum length.

Ethernet implementations shall support at least one of the following maximum MAC Client Data field sizes:

- 1,500 decimal octets – standard (basic) frame
- 1,504 decimal octets – Q-tagged frame
- 1,982 decimal octets – envelope frame

This field directly carries the information from the MAC client, which will subsequently be followed by padding (if required) and the Frame Check Sequence (FCS) for error detection.

VLAN and Q-Tagged Frames

In Ethernet, **VLANs (Virtual Local Area Networks)** are used to partition a single physical network into multiple logical networks. This strategy is implemented at the Data Link Layer (Layer 2) of the ISO/OSI model. Each host is assigned a **VLAN ID**, which allows switches to segregate network traffic according to security policies or to separate traffic belonging to different logical domains or zones.

A **Q-tagged frame** includes an additional 4-octet field that specifies the VLAN from which the frame originated.

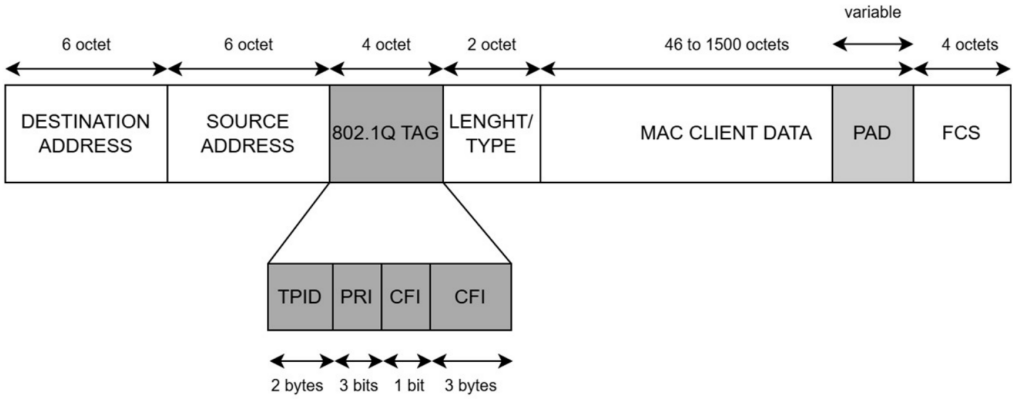

Figure 2.15: Qtagged frame structure

- **TPID (Tag Protocol Identifier)**: Indicates the type of frame. The standard value `0x8100` identifies an IEEE 802.1Q frame. Devices not supporting 802.1Q discard these frames. Vendors may define their own TPID values for interoperability between devices from different manufacturers.
- **PRI (Priority)**: Specifies the frame priority, ranging from 0 to 7. Higher values indicate higher priority messages.
- **CFI (Canonical Format Indicator)**: Indicates whether the MAC address is encapsulated in canonical format, ensuring compatibility between Ethernet and Token Ring networks.
- **VID (VLAN Identifier)**: Identifies the VLAN to which the frame belongs. Valid values range from 0 to 4095, with the first and last values reserved.

Padding

A **minimum MAC frame size** is required for correct protocol operation. If the MAC Client Data field is smaller than the minimum, a **Pad field** is appended immediately after the data field, before the **Frame Check Sequence (FCS)** is added.

The size of the Pad field is determined by the length of the MAC Client Data field. For a MAC Client Data field of length `clientDataSize` octets, the required Pad length (in bits) is calculated as:

$$\text{Pad length} = \max\left[0, \text{minFrameSize} - (\text{clientDataSize} + 2 \times \text{addressSize} + 48)\right]$$

This ensures that the total frame length meets the minimum requirement for proper operation of the protocol.

Frame Check Sequence (FCS)

The **CRC (Cyclic Redundancy Check)** is a value used to verify the integrity of a MAC frame. It is included in the 4-octet **FCS field** at the end of the frame.

The CRC is computed by the sender and attached to the frame. Upon reception, the receiver recomputes the CRC over the received data and compares it to the received FCS. If the computed and received CRC values differ, the frame has been altered during transmission, and its content is considered unreliable.

The FCS allows the detection of symbol errors and ensures that only valid frames are processed by the receiving device.

CRC Calculation

The basic idea behind any **CRC (Cyclic Redundancy Check)** algorithm is straightforward: the data stream is treated as a large binary number, which is divided by a fixed (constant) binary number, and the remainder of this division is used as the CRC value.

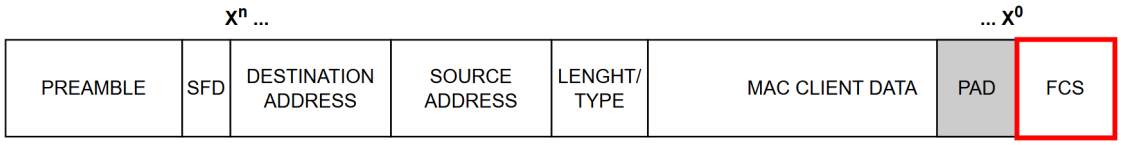

Figure 2.16: CRC calculation field with bit sequence order

The CRC is computed as a function of the contents of the protected fields of the MAC frame:

- Destination Address
- Source Address
- Length/Type field
- MAC Client Data
- Pad

Mathematically, the CRC value for a given MAC frame is calculated using the following procedure:

1. The first 32 bits of the frame are complemented.
2. The n bits of the protected field are considered as the coefficients of a polynomial $M(x)$ of degree $n-1$.
3. $M(x)$ is multiplied by x^{32} and divided by the generator polynomial $G(x)$, producing the remainder $R(x)$ of degree ≤ 31.
4. The coefficients of $R(x)$ are treated as a 32-bit sequence.
5. The bit sequence is complemented; the result is the final CRC.

The polynomials used in the calculation are defined as follows:

$$\mathbf{M(x) = x^{32} + x^{31} + \cdots + x^{22} + x^{21} + \cdots + x^{12} + x^{11} + \cdots + x^5 + x^4 + x^3 + x^2 + x + 1}$$

$$\mathbf{G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1}$$

$$\mathbf{R(x) = \frac{M(x) \cdot x^{32}}{G(x)}}$$

$$\mathbf{CRC = R(x)^{-1}}$$

This method ensures that any alteration in the protected fields during transmission is detected, allowing the receiver to identify corrupted frames reliably.

End Frame Delimiter (EFD)

The **End Frame Delimiter**2.5.1 is a signal indicating the completion of a frame reception. It is important to note that the EFD is not a numerical value embedded within the frame itself.

Instead, it is represented by the **deassertion of the RX_DV (Receive Data Valid)** control signal. This deassertion marks the end-of-frame for the data transmitted on the RXD[3:0] bus, signaling to the receiver that the complete frame has been received and processing can proceed.

Error Detection Analysis

100BASE-T1, being based on Ethernet, implements a 32-bit CRC as a parity check to detect possible errors such as bit flips during message transmission.

The **undetected error probability** P_{ue} is the probability that an error occurs during transmission but cannot be detected by the decoder. This can happen in the following cases:

- An error pattern transforms a given codeword c into a different codeword $c' = c + e$. This occurs only if the error pattern itself forms a valid codeword.
- Channel noise produces an error pattern equal to a non-zero codeword of the code.

Mathematically, this probability arises because an undetected error pattern corresponds to one of the 2^{32} possible codewords resulting from the CRC generation.

$$\rho = \mathbf{2^{-32} \approx 2.3 \times 10^{-10}} \tag{2.5}$$

Formula 1: Probability of non-detected faulty Ethernet standard frames.

$$\mathbf{BER < 10^{-10}} \tag{2.6}$$

Formula 2: Bit Error Rate (BER) allowed.

$$\mathbf{FER < 10^{-7}} \text{ for 125 octets} \tag{2.7}$$

Formula 3: Frame Error Ratio (FER).

CRC Error

With **Cyclic Redundancy Check (CRC)**, the transmitter calculates a checksum over the CRC bit sequence, starting from the **Start of Frame** bit up to the end of the **Data Field**.

A **CRC error** is detected if the CRC calculated by the receiver does not match the received CRC. In this case, the frame is considered invalid and must be discarded.

It is important to note that the CRC checksum is used solely for **error detection**. Error correction is not performed, as it is not implemented in this protocol.

Hamming Distance and Error Correlation

The **Hamming Distance (HD)** represents the minimum number of bit changes required to transform one valid codeword into another. In the context of Ethernet, the implemented polynomial for CRC calculation in **100BASE-T1** guarantees a minimum Hamming distance of **HD = 4** for a given frame length range.

This means that no combination of 1-, 2-, or 3-bit errors can result in an undetected error, while there exists at least one possible 4-bit error pattern that may not be detected. Consequently, the CRC implemented in 100BASE-T1 is capable of detecting up to **three independent bit errors** within an Ethernet **Maximum Transmission Unit (MTU)** of 1500 bytes.

However, using more complex generator polynomials, it is theoretically possible to reach higher Hamming distances (up to **HD = 6**), thus detecting up to five independent bit errors. It is important to note that both the undetected error probability (P_{ue}) and the effective HD depend on the message length: longer frames tend to have a smaller minimum Hamming distance, reducing error detection capability.

Table 2.4: Hamming distance correlation with codeword length.

Code Length n [bit]	**Minimum Hamming Distance $d_{\mathbf{min}}(n)$ [bit]**
3007 – 12144	4
301 – 3006	5
204 – 300	6
124 – 203	7
90 – 123	8
67 – 89	9
54 – 66	10

Jabber Error Detection

The **Jabber Error Detection** function is implemented to prevent a PHY from remaining locked in the *DATA* state of the Physical Coding Sublayer (PCS). This situation can occur when the *End-of-Frame Delimiters* (ESD1, ESD2) are not correctly detected, causing the receiver to stay indefinitely in the data reception phase.

To avoid this condition, a **Jabber timer** is introduced, set to a duration of **1.08 ms ± 54 µs**. This represents the maximum time a receiver is allowed to remain in the DATA state. If this timer expires, or if the maximum message length is exceeded, the PHY is forced to return to the **IDLE** state.

This mechanism ensures network stability by forcing both nodes to attempt a new link establishment, allowing recovery from transmission errors or synchronization loss that may

have caused the faulty condition.

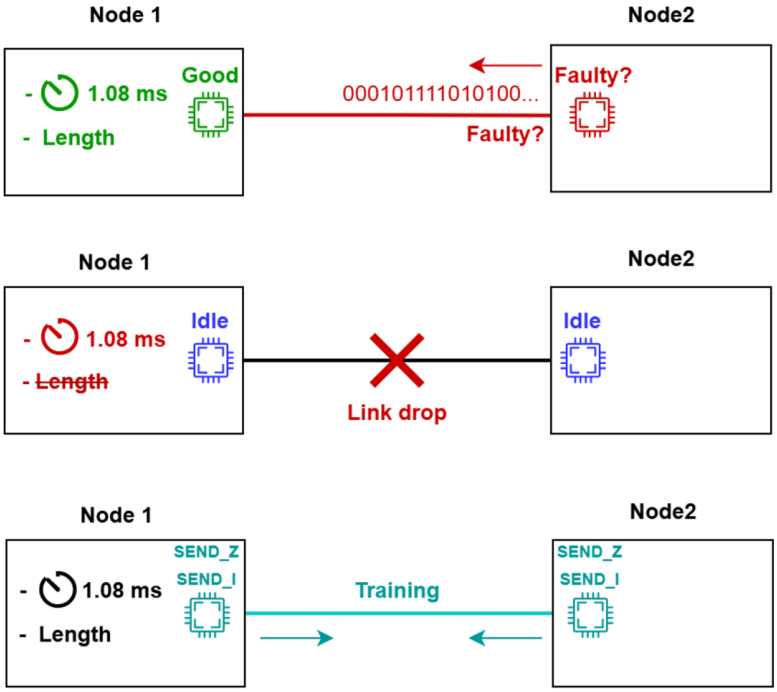

Figure 2.17: Jabber Error identification.

CHAPTER 3

Ethernet Automotive Application Protocols

On top of the basic physical and data link layers, several higher-layer protocols have been developed to enable communication, synchronization, and service-oriented interactions between different Electronic Control Units (ECUs).

These protocols define how data are structured, transmitted, and interpreted within the vehicle network, depending on the purpose of communication. They are essential to ensure interoperability between devices from different suppliers, to guaranty real-time performance when required, and to provide deterministic and synchronized data exchange.

In this chapter, the main Automotive Ethernet communication protocols are introduced and described:

- **DoIP (Diagnostics over IP)** – allows diagnostic services to be performed over standard IP-based Ethernet networks.
- **SOME/IP (Scalable service-Oriented Middleware over IP)** – a middleware protocol enabling service-oriented communication between ECUs.
- **AVB (Audio Video Bridging)** – a set of IEEE standards providing time-synchronized and low-latency transmission of audio and video streams.
- **TSN (Time-Sensitive Networking)** – the evolution of AVB, extending determinism and time awareness to a broader range of applications.
- **gPTP (generalized Precision Time Protocol)** – a precise time synchronization protocol based on IEEE 802.1AS, used to align the local clocks of all network nodes.

Each of these protocols addresses a specific communication need in the automotive ecosystem, from infotainment and diagnostics to control and time synchronization. Their coordinated use allows modern vehicles to achieve high-performance, flexible, and reliable Ethernet-based communication architectures.

3.0.1 DoIP

Vehicle Diagnostics has been developed starting by the need to take measurements in order to determine malfunctions effect in vehicles. Now it is evolved to cover various field case such as emission-data logging or calibration and update of ECUs. Nowadays increasing need to update of software and analyzing more and more number of function provided by the ECUs, have increase the complexity of the automotive network making it similar to normal LAN.[6] **ISO 13400** was created for this purpose, to define a common requirements to develop a system implemented diagnostic protocols exploiting the advantages given by an IP network. The main feature descripted are:

- Separate the in-vehicle network from the diagnostic DoIP entity interface requirements.
- Allow long-term stable vehicle communication used to interface the in-vehicle network with an external host.
- Reuse and update industrial standards that use long-term stable communication already legislated for diagnostic communication, to meet also the manufacturer-specific use cases.
- Easy adaptation to all physical and data link layers already present and make it simple to future new protocols adaptation.

The vehicle diagnostic communication framework of the DoIP is developed crossing all the 7 layers of the ISO-OSI standard, each layer function is defined by a precise standard that assure all the functionality of the DoIP framework protocol.

Table 3.1: Protocol and Standard Mapping for DoIP Communication Stack

OSI Layer	**Standard / Protocol**	**Function in DoIP Context**
Application (Layer 7)	**ISO 14229-1 / ISO 14229-5 (UDS on IP)**	Defines diagnostic services and communication mechanisms for vehicle ECUs over IP. Implements Unified Diagnostic Services (UDS) on TCP/IP.
Presentation (Layer 6)	**ISO 22901** (ODX - Open Diagnostic Data Exchange)	Specifies data format and parameter descriptions used for diagnostic communication. Enables standardized representation of diagnostic data.
Session (Layer 5)	**ISO 14229-2** (Session Layer Service Interface)	Defines session management and timing for diagnostic communication. Provides the upper layer service interface for establishing and maintaining diagnostic sessions.
Transport & Network (Layers 4 & 3)	**ISO 13400-2** (Transport Layer and Network Layer Services) **TLS / TCP / UDP / IP**	Specifies message segmentation, transmission, and addressing mechanisms for DoIP. Ensures reliable data transfer (TCP) or low-latency communication (UDP) over IP networks.
Data Link (Layer 2)	**ISO 13400-3** (DoIP – Part 3) **IEEE 802.3** (Ethernet)	Defines physical and data link specifications for wired Ethernet-based **DoIP** communication. Provides frame transmission, addressing, and synchronization mechanisms.

Opening of TCP channel

The establishment of a *Diagnostic over IP* (DoIP) communication session begins with the creation of a **TCP socket connection** between the external test equipment (tester) and the **DoIP entity** implemented within the vehicle's ECU gateway. This connection setup

must be completed prior to any diagnostic message exchange, as it provides the logical channel for routing and session control over the underlying **ISO 13400-2** transport layer.

Once the TCP connection is successfully established, the **initialization phase** is executed. During this phase, both the *initial inactivity timer* and the *general inactivity timer* are started to monitor the communication session according to DoIP timing constraints defined in ISO 13400-2 2019.

Routing activation

To enable message routing through this connection, the tester sends a **Routing Activation Request Message** to the DoIP entity. This message requests permission to register the connection for diagnostic communication. If the external tester is considered *eligible*—meaning it meets the authentication, addressing, and network configuration criteria defined by the DoIP implementation—and the number of active connections does not exceed the maximum allowed value N, the DoIP entity accepts the request. The initial inactivity timer is then stopped, and the connection state transitions to **"registered"**, indicating that the routing path is now active.

The DoIP entity confirms successful activation by transmitting a **Routing Activation Response Message** with a positive acknowledgment code. From this point, valid DoIP payloads can be routed through the active session. The *general inactivity timer* is restarted to continue monitoring link activity.

Exchange of Diagnostic message

When a new data frame is received, the DoIP entity first invokes the **DoIP Header Handler**, responsible for verifying and parsing the DoIP header structure. If the payload corresponds to a diagnostic message, the frame is passed to the **Diagnostic Message Handler**—typically implementing **UDSonIP** as defined in ISO 14229-5—for higher-layer processing.

Upon successful reception and validation of a diagnostic message, the DoIP entity sends a **Diagnostic Message Confirmation (ACK)** to the tester, signaling that the message has been correctly received and processed by the diagnostic handler. If the payload contains a UDS-compliant request, the addressed ECU generates a diagnostic response that is transmitted back to the tester through the same TCP socket.

Closure of TCP channel

When diagnostic communication is completed, the external test equipment should properly close the connection using standard **TCP termination procedures**. The DoIP entity then executes the **finalization sequence**, releasing allocated resources and freeing the socket for potential new sessions. If the connection is not explicitly closed, cleanup is automatically triggered after the *general inactivity timeout* expires or following a connection status verification routine, ensuring that system resources are correctly managed and preventing potential session lock states.[16]

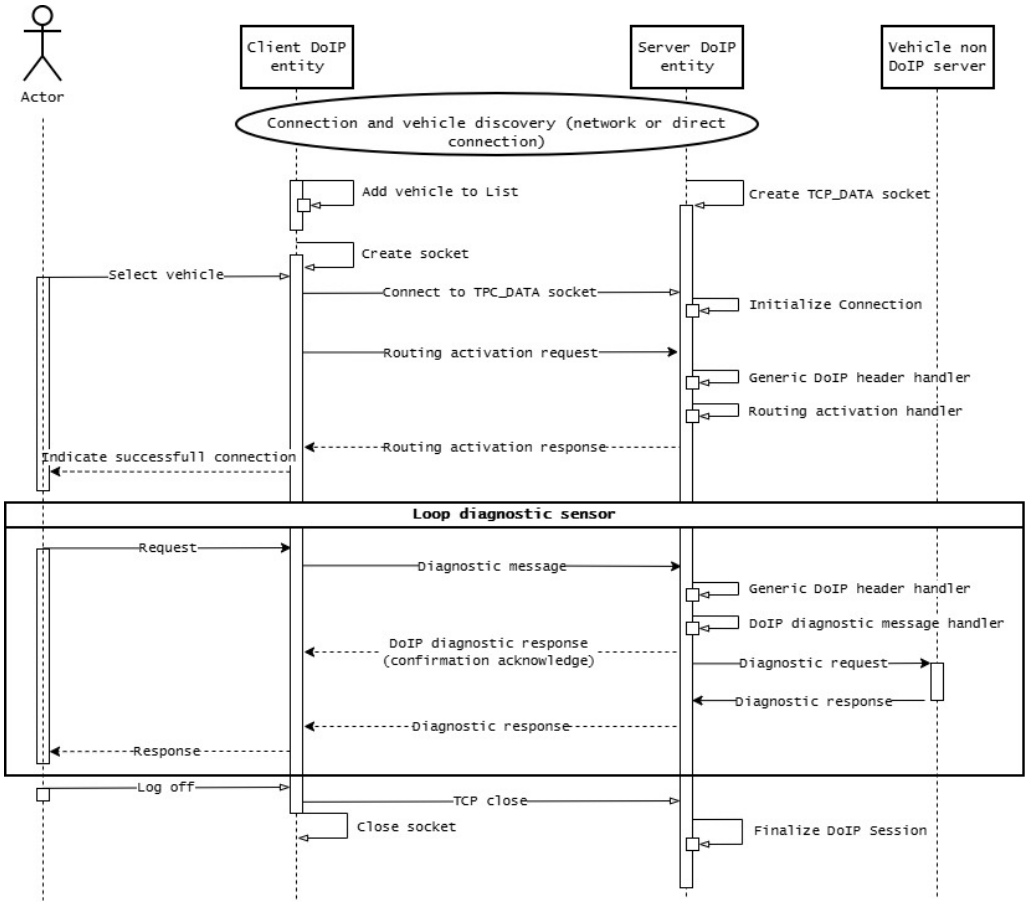

Figure 3.1: The DoIP functional scheme highlights how diagnostic communication is handled over TCP/IP networks, enabling efficient routing, session management, and secure data exchange between external test equipment and in-vehicle ECUs. Figure rielaboration from [6]

In conclusion, the DoIP protocol represents the natural evolution of vehicle diagnostics towards Ethernet-based architectures. By leveraging IP communication, it provides higher bandwidth, scalability, and integration with modern IT infrastructures. Compared to traditional CAN-based diagnostics, DoIP enables faster data exchange, supports remote operations, and ensures greater flexibility for future connected vehicle platforms.

3.1 SOME/IP Middleware

SOME/IP (Scalable service-Oriented Middleware over IP) is an automotive communication protocol designed to enable service-oriented communication over Ethernet networks. It is part of the AUTOSAR standard and provides a flexible middleware for real-time and high-bandwidth applications such as ADAS sensors, infotainment, and vehicle diagnostics.

Unlike classic fieldbus communication based on static message identifiers, SOME/IP is based on dynamic discovery and service invocation models similar to modern IT systems.

3.1.1 Communication Model

SOME/IP supports two fundamental interaction patterns:

- **Service Discovery (SOME/IP-SD)**: Clients dynamically locate and subscribe to services provided by servers on the network. Advertisements include:

  - Service IDs
  - Instance IDs
  - Endpoint information (IP, port)

- **Service Invocation**: Once discovered, services can be used via request/response or event-based communication:
  - **Request/Response**: synchronous or asynchronous message exchange
  - **Event groups**: push-based update of data to subscribed clients

3.1.2 SOME/IP Packet Structure

SOME/IP payload is transported over UDP or TCP. A SOME/IP message contains the following fields:

Field	**Description**
Service ID	Identifies the provided service
Method ID / Event ID	Operation or event triggered
Length	Message size including payload
Client ID	Sender identifier
Session ID	Used for matching request/response
Protocol Version	SOME/IP protocol version
Interface Version	Supported service interface version
Message Type	Request/Response/Event semantics
Return Code	Status of the operation
Payload	Application-specific data

Table 3.2: Structure of a SOME/IP message header.

3.1.3 Advantages in Automotive Networks

- Enables distributed services in a flexible and scalable architecture
- Integrates seamlessly with Ethernet and IP standards
- Supports dynamic topology changes through discovery mechanisms
- High bandwidth suitable for sensor-rich autonomous systems

For these reasons, SOME/IP plays a key role in service-oriented vehicle architectures where applications running on different ECUs cooperate through network-based interfaces rather than static signal routing.

3.2 Time Sensitive Networking (TSN)

Time Sensitive Networking (TSN) is a set of IEEE 802.1 standards designed to enable deterministic communication over standard Ethernet networks. The main goal of TSN is to guarantee bounded latency, extremely low jitter, and minimal packet loss in applications such as automotive, industrial automation, and audio/video transmission.

Traditional Ethernet operates on a best-effort basis and does not provide any timing guarantees. TSN extends Ethernet by adding scheduling and synchronization functions so that time-critical traffic can safely coexist with conventional best-effort traffic.

3.2.1 Key Features of TSN

TSN introduces several mechanisms to ensure deterministic behavior:

- **Precise Time Synchronization**: Achieved using gPTP (IEEE 802.1AS), which enables a common clock across the network with sub-microsecond accuracy.
- **Time-Aware Traffic Scheduling**: Defined in IEEE 802.1Qbv, where time slots are reserved for specific traffic classes. Frames can be sent only within their assigned window.
- **Traffic Shaping and Policing**: Standards such as IEEE 802.1Qav introduce credit-based shaping to control outgoing traffic and maintain latency guarantees.
- **Frame Preemption**: IEEE 802.1Qbu allows interrupting low-priority frames to transmit high-priority traffic immediately, reducing blocking delays.
- **Reliability and Redundancy**: IEEE 802.1CB enables seamless redundancy with no frame loss during link failures by duplicating and merging streams.[17]

These features allow deterministic communication without requiring a separate fieldbus network, reducing wiring complexity and cost in automotive architectures.

3.2.2 TSN Stream Model

TSN traffic is structured into *streams*. Each stream is identified and handled according to a predefined configuration stored in every switch along the path:

- **Stream Identification**: Based on MAC address, VLAN tag, and priorities
- **Stream Reservation**: Managed through IEEE 802.1Qcc and SRP (Stream Reservation Protocol)
- **Bounded Latency**: Guaranteed by allocating bandwidth and scheduling transmission windows

3.2.3 Frame Format

TSN reuses the standard Ethernet frame format with additional fields when needed, such as:

- **VLAN Tag (IEEE 802.1Q)** including the Priority Code Point (PCP) to classify traffic
- **Sequence number fields** for redundancy (802.1CB)

Thanks to these mechanisms, TSN enables the convergence of real-time control traffic, infotainment, and best-effort communication on the same physical network, which is a fundamental requirement for modern automotive E/E architectures.

3.3 Generalized Precision Time Protocol (gPTP)

Generalized Precision Time Protocol (gPTP) is defined in the IEEE 802.1AS-2020[18] standard, its primary objective is to distribute a common notion of time among all devices in a local network, enabling deterministic and synchronous operations — a fundamental requirement in automotive and industrial environments.

In gPTP, each device participating in the protocol exposes several functional entities that cooperate to achieve accurate time synchronization across the network. The most important ones are briefly described below:

- **PTP Instance**: The software component running inside a network device that implements the gPTP protocol logic. If the device is not elected as Grandmaster, the PTP instance acts as a slave and adjusts its local clock based on the information received.
- **PTP Relay Instance**: When a node is positioned between the Grandmaster and other devices, its PTP relay instance receives synchronization information from the upstream node, compensates for local path delays, and then forwards the corrected timing information to downstream nodes. This enables multi-hop synchronization.
- **PTP Port**: Logical interface used by a PTP instance to exchange synchronization messages. Each physical network interface may include one or more PTP ports.
- **Local Clock Entity**: Hardware timebase of a node. Its stability strongly influences the global synchronization accuracy. Some automotive Ethernet PHYs (e.g. 100BASE-T1 transceivers) integrate a Real-Time Clock directly in hardware, improving jitter performance and reducing clock drift.

In a gPTP-enabled network, one node is elected as the *Grandmaster* (GM), which owns the most accurate and stable clock. The GM periodically distributes timing information to the other network nodes, which continuously adjust their local clocks considering both the received time data and the propagation delay introduced by the communication path.

3.3.1 Best Master Clock Algorithm (BMCA)

Each gPTP-capable device implements the Best Master Clock Algorithm (BMCA), which selects the Grandmaster based on a comparison of several clock quality parameters, such as clock class, accuracy, stability, and priority fields. If a better clock becomes available (e.g. a new device joins or a clock degrades), the BMCA triggers a re-election, and the network switches seamlessly to the new Grandmaster.

BMCA ensures:

- Continuous availability of a reliable time reference
- Automatic recovery from node failure or network topology changes
- Only one active GM within the domain at any time

Each device has one or more *PTP ports* that exchange synchronization messages and a *LocalClock*; the latter determines synchronization accuracy, so good oscillator stability is highly beneficial.

3.3.2 PTP Message Types in gPTP

All messages used in gPTP are defined by IEEE 802.1AS-2020 and can be grouped into two major categories: timestamped *Event messages* and untimestamped *General messages*.

Message type	Value
Sync	0x0
Pdelay_Req	0x2
Pdelay_Resp	0x3
Follow_Up	0x8
Pdelay_Resp_Follow_Up	0xA
Announce	0xB
Signalling	0xC

Table 3.3: PTP message type encoding (IEEE 802.1AS-2020).

Event Messages

Event messages are timestamped on transmission and reception. They directly contribute to the computation of clock offset and path delay.

- **Sync**: distributes the Grandmaster timestamp. In *two-step mode* the precise timestamp is carried by a subsequent Follow_Up message.
- **Pdelay_Req** and **Pdelay_Resp**: used in the peer-to-peer delay mechanism to measure link delay between two adjacent nodes.

General Messages

These messages support clock management and do not carry timestamps.

- **Announce**: contains Grandmaster identity and quality parameters used by BMCA.
- **Signalling**: carries configuration or timing information between nodes.
- **Follow_Up**: contains the precise GM timestamp associated with a Sync message.

3.3.3 Propagation Delay Measurement (Peer-to-Peer)

gPTP uses the peer-to-peer (P2P) mechanism to estimate the *mean propagation delay* of each link. The measurement is performed independently by both ends of a full-duplex link and is continuously refreshed, which allows quick resynchronization after topology changes.

The computation uses four timestamps:

- t_1: Pdelay_Req transmission egress timestamp at the requester
- t_2: Pdelay_Req reception ingress timestamp at the responder
- t_3: Pdelay_Resp transmission egress timestamp at the responder
- t_4: Pdelay_Resp reception ingress timestamp at the requester

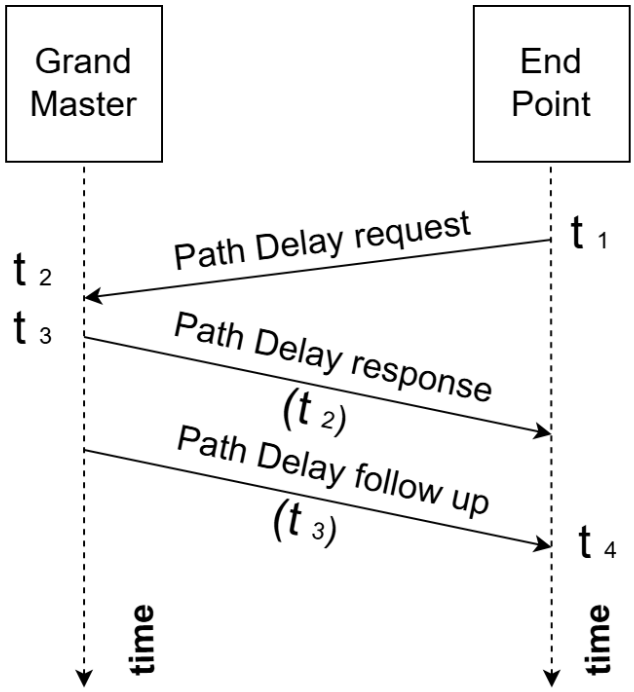

Figure 3.2: Pdelay calculation message exchange

The responder includes t_2 in the Pdelay_Resp and t_3 in the Pdelay_Resp_Follow_Up message. The requester computes:

$$\text{meanLinkDelay} = \frac{(t_4 - t_1) - (t_3 - t_2)}{2} \tag{3.1}$$

This value represents the average propagation time of the link. If the clocks at both ends have different rates, an additional rate compensation factor is applied (not shown here for clarity).

3.3.4 Synchronization and Clock Adjustment

Synchronization is maintained by periodically exchanging Sync (and optionally Follow_Up) messages from the Grandmaster to all other nodes.

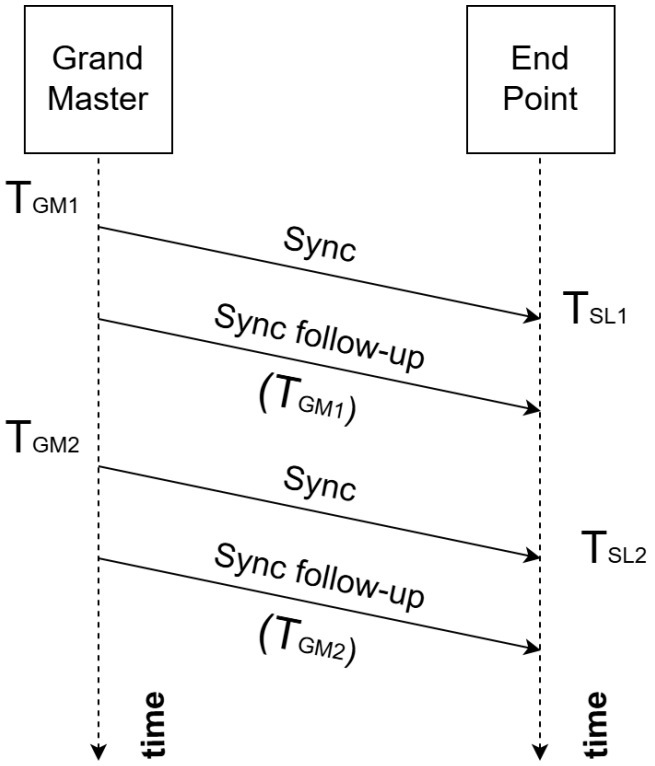

Figure 3.3: Periodically sync message sended by the master to the slaves

- The GM timestamps transmission at t_{GM}
- The slave timestamps reception at t_{SL}

Once the slave knows:

- the precise origin timestamp from the GM (via Sync or Follow_Up)
- the propagation delay of the path (from Pdelay)

It can compute the *offset* between its LocalClock and the GM clock:

$$\text{offset} = (t_{SL} - t_{GM}) - \text{meanLinkDelay} \tag{3.2}$$

A clock servo (typically a PI controller) continually adjusts the local oscillator in both phase and frequency so that the offset tends toward zero. If the GM becomes unavailable, the slave enters a *holdover mode*, slowly drifting according to its own oscillator stability.

CHAPTER 4

Experimental Setup and Measurement Infrastructure

4.1 Introduction

The objedtive of this work is to implement and test a basic point to point network with the 100BASE-T1. As Ethernet Automotive with TSN and AVB aim to carry big frames of data, synchronization of the network is one of the essential functionality to be assure. For this reason the implementation of the testing network comprends the adaption of the NXP gPTP stack to work directly with the lower RTL drivers. In the following chapters describes the hardware and software set up and the experimental tests made to validate and test the synchronization of the network in different condition.

4.2 Automotive Ethernet Support Infrastructure

100BASE-T1 Pre-Analysis Using Media Converters and TAP Devices

Before deploying the protocol stack on the embedded boards, preliminary measurements and traffic inspection were required to understand how 100BASE-T1 behaves in practice. In order to simulate a 100BASE-T1 network some testing devices are crucial to understand the architecture and the data flow of the frame message.

Media Converters

The primary difference between conventional Ethernet networks—typically deployed in standard LAN environments using RJ45 connectors—and Automotive Ethernet based on the 100BASE-T1 standard lies in the physical medium and connector technology. Standard Ethernet employs a *four-pair unshielded twisted-pair (UTP)* cable, whereas 100BASE-T1 uses a *single balanced twisted pair* specifically engineered for the automotive environment, providing reduced weight, lower cost, and improved electromagnetic robustness.

A media converter bridges these two physical layers by translating between classical Ethernet and 100BASE-T1 signalling. In practical terms, it enables a device equipped with a standard RJ45 Ethernet interface to communicate over an automotive single-pair link. As a result, the media converter is an essential tool when simulating or emulating an automotive network node.

These devices are particularly valuable in laboratory settings, test benches, and rapid-prototyping workflows. For example, they allow engineers to observe and analyse the actual traffic generated by an Electronic Control Unit (ECU) using standard PC-based tools, or to emulate an ECU using a conventional computer equipped with a standard

Ethernet network interface card. By providing the appropriate *Physical Layer (PHY)* and *Data Link Layer* translation, media converters integrate seamlessly into a 100BASE-T1 network while maintaining compatibility with traditional Ethernet infrastructure.

FibreCode – FC602 This device integrates a 100BASE-T1 Physical Layer and exposes a virtual network interface to the host PC through USB. It was primarily used during the early stages of the experimentation to capture and inspect traffic using Wireshark. Its integrated NIC allowed direct packet logging on the host machine. It's Python api functionality allows to access low register level information of the phy making it suitable for physical-level debugging and monitoring.

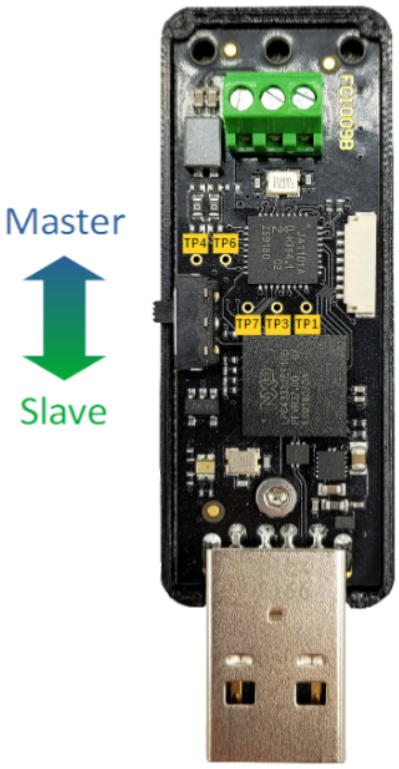

Figure 4.1: MDI of FC602 device are 2 cable clamp with screw to attach TP+ and TP- cables. Source: [7]

Technica Engineering – MEDIACONVERTER 100/1000BASE-T1 This converter provides a standard RJ45 Ethernet interface and exposes hardware dip-switches to select between 100BASE-T1 and 1000BASE-T1 operation, as well as to manually configure the Master/Slave role on the single-pair link. In the tests, it was set to operate in 100BASE-T1 mode with a fixed Master or Slave role, depending on the scenario under investigation. Its debug port expose a shell from where it is possible to access to frame statistics or register reading and writing.

Figure 4.2: Cable connectors for the 100BASE-T1 side are choosen to be HMTD, compliant to most automotive network solutions. Source: [8]

Intrepid Control System – RAD Moon Unlike the previous device, this converter does not include physical switches. Instead, it automatically determines whether to operate as Master or Slave based on link training phase. It was employed in scenarios where rapid reconfiguration was needed, or when testing the behaviour of the S32K344 boards with a converter that does not impose a predefined role.

Figure 4.3: Cable connector for input data stream is Molex type. The two led assure controllability on link establishment and activity on the medium. Source: [9]

Ethernet TAP Devices

Ethernet differs fundamentally from bus-oriented communication technologies such as CAN, as it relies on a point-to-point, full-duplex link architecture. Consequently, a passive "listener" placed on the cable cannot observe all traffic on the medium, since no shared

broadcast channel exists. To enable non-intrusive traffic inspection in such networks, *Test Access Point* (TAP) devices are employed.

An Ethernet TAP integrates two or more Ethernet PHYs couples that emulate the endpoints of the monitored link. By establishing two independent full-duplex connections—one towards each real endpoint—the TAP can capture every frame traversing the link. Internally, an FPGA acquires the frames, replicates them with minimal and deterministic delay, and forwards them to one or more dedicated monitoring ports. These ports can be connected to tools such as Wireshark or hardware traffic analyzers.

At the same time, all traffic received on each PHY is transparently reinjected into the opposite side of the link, ensuring that the network under test operates exactly as if no TAP were present. As they receiving all the traffic of the network they can be also used as filter medium before a datalogger in order to filter or eventually inject data frames. Thanks to this transparency and deterministic behaviour, TAP devices are indispensable in automotive Ethernet test setups, conformance testing, and debugging scenarios where full visibility of the communication is required.

Star Electronics - FL3X TAP The *FL3X TAP* is a multi-channel Automotive Ethernet TAP designed to monitor up to six independent 100BASE-T1 links simultaneously. The device integrates twelve Ethernet PHYs arranged in paired full-duplex configurations, enabling complete acquisition of the traffic flowing on each monitored link. All captured frames are timestamped in hardware, providing deterministic and high-precision temporal alignment across channels.

The recorded traffic is exported through one of the two available logging ports, which also serve as configuration interfaces. Each port supports an output bandwidth of up to 1 Gb/s, ensuring lossless streaming of high-volume traffic typical of automotive test environments.

In this experimental campaign, the Fl3x TAP was used to observe and analyze the raw traffic exchanged within the test network. Its ability to transparently capture, timestamp, and relay Ethernet frames made it a crucial instrument for debugging, performance evaluation, and rapid prototyping.

Figure 4.4: Cable connectors for input data stream are HMTD type. Front LEDs assure the link establishment and data flowing through the PHYs and the logging ports. Source: [10]

4.3 Embedded Hardware Platform

4.3.1 Evaluation Boards

The experimental setup requires the use of two evaluation boards equipped with 100BASE-T1–capable PHYs. The most suitable candidates are the S32K344-CANHUB and the S32K344-WB boards. Both platforms integrate the same microcontroller, the S32K344, in different packages but with an identical architecture.

The S32K344 is a 32-bit microcontroller based on an Arm Cortex-M7 architecture and features a dual **lockstep** core configuration for functional safety. This type of MCU is widely adopted for rapid prototyping in automotive applications because many key peripherals are natively integrated into the device. In particular, it includes:

- one Ethernet MAC (GMAC) supporting bandwidths up to 100 Mbps and capable of software timestamping, which is essential for real-time and time-sensitive networking;
- a system clocking unit based on an internal PLL, using an external oscillator as reference, allowing the MCU to operate up to 160 MHz. This clocking infrastructure also provides a stable basis for synchronization and clock correction mechanisms;
- low-latency timers such as the PIT, used for task scheduling and performance profiling;
- a high-speed SERDES RGMII interface enabling connection between the MCU and an external PHY through the Reduced Gigabit Media Independent Interface;
- hardware safety features, including the watchdog interface provided by NXP's FS26;
- a comprehensive set of automotive communication interfaces, such as UART, LIN, I^2C, SPI, CAN, CAN-FD, and FlexCAN.

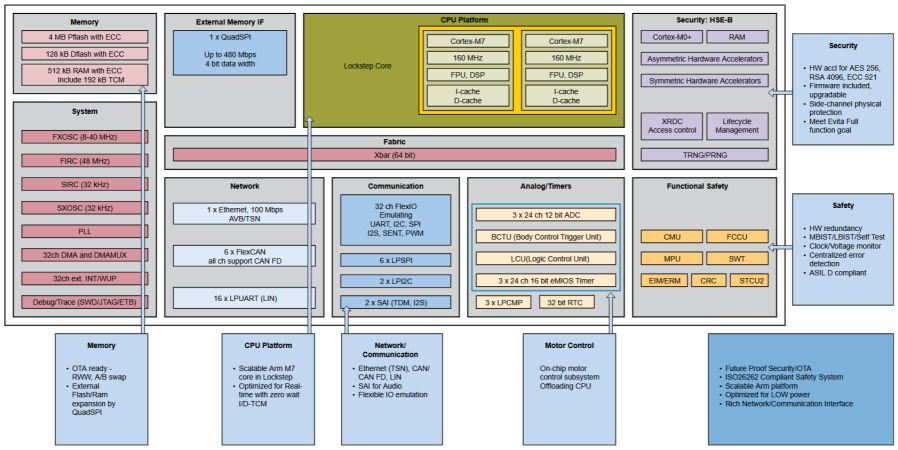

Figure 4.5: Internal block diagram of the S32K344 microcontroller. Source: [11]

MR-CANHUBK344

This evaluation board was chosen because it integrates a 100BASE-T1 PHY, specifically the NXP TJA1103. This device is a single-port transceiver capable of establishing communication with another node in a 100BASE-T1–based network. The TJA1103 represents the third-generation 100BASE-T1 PHY developed by NXP [19], and it is designed to meet ASIL-B functional safety requirements.

Furthermore, the PHY is compliant with IEEE 802.3 automotive Ethernet specifications and supports time-synchronization protocols such as gPTP. For this purpose, it includes a hardware timestamping unit that enables accurate ingress/egress timestamp acquisition with minimal latency. This feature provides significantly more precise timing information compared to software-based timestamping.

For these reasons, this evaluation board was selected to operate as the Grand Master in the experimental network setup, as the hardware-assisted timestamping yields higher precision and stable timing informations.

Component	**Type**	**Role in the experiment**
MCU	NXP S32K344	Main processing unit, running gPTP stack and traffic generation.
PHY	NXP TJA1103	100BASE-T1 transceiver with hardware timestamping support.
Clock source	External oscillator	Provides reference clock for the MCU PLL and for synchronization functions.
Watchdog	NXP FS26	Power management and safety monitoring.
Ethernet Interface	RGMII	High-speed interface between MCU GMAC and PHY.

Table 4.1: Main components of the S32K344-CANHUB board used in the experimental setup

S32K344-WB Board

The NXP S32K344-WB "White Board" is a general-purpose automotive evaluation platform designed for rapid prototyping and early development of in-vehicle networking applications. It integrates all the key peripherals commonly used in automotive ECUs, including CAN FD, LIN, UART, SPI, I^2C, and both 100BASE-T1 and 100BASE-TX Ethernet interfaces. The board employs the 257-pin BGA package of the S32K344 microcontroller, ensuring high flexibility in routing signals to the integrated network components.

The board features multiple 100BASE-T1 transceivers:

- **TJA1101**: single-port 100BASE-T1 PHY
- **TJA1102**: dual-port 100BASE-T1 PHY

The most relevant component for this thesis is the **SJA1105QS automotive Ethernet switch**. This 5-port switch supports frame filtering and forwarding based on header fields (e.g., EtherType, VLAN tags), and provides connectivity for 10/100/1000 BASE-T1/TX and even optical links. It is fully compliant with AVB/TSN features and supports hardware-assisted time synchronization, which makes it particularly suitable for experiments involving gPTP and time-sensitive networking.

Table 4.2: Main hardware features of the S32K344-WB evaluation board.

Component	Description
MCU	NXP S32K344, ARM Cortex-M7 dual-core, 257-pin BGA package
Ethernet PHYs	• TJA1101 – single-port 100BASE-T1 • TJA1102 – dual-port 100BASE-T1
Ethernet Switch	SJA1105QS, 5-port automotive Ethernet switch, supports VLANs, QoS, TSN/AVB features, forwarding and filtering rules
Connectivity	100BASE-T1, 100BASE-TX, CAN FD, LIN, UART, SPI, I^2C
Clocking	External oscillator feeding MCU PLL; suitable for synchronized networking applications
Debug	Integrated JTAG / SWD interface

4.3.2 Programming and Debugging Tools

S32 Design Studio

The development and debugging features were done using the NXP S32 Design Studio (S32DS). S32DS IDE based on Eclipse integrates the GCC toolchain for compiling and debugging, configuration wizards, memory tool inspection and native development environment for all NXP's microcontrollers. It is used to integrates all the Real-Time drivers(RTD), which were used in this project. All firmware was generated, compiled and deployed by the S32DS version 5.

Firmware flashing and real-time debugging were performed using the Segger J-Link probe using a SWD access. This made simple the usage of breakpoint to inspect code and memory locations via non-intrusing real-time debugging. The J-link is fully supported by S32DS and allows stable and deterministic interaction with the S32K344 MCUs during development and debugging sessions.

4.4 Network Architecture and Software Implementation

After selecting the two evaluation boards, the next step was to define the test network and decide which device should operate as the Grand Master and which as the Slave. This choice was mainly driven by the clock-correction capabilities available on each platform, since maintaining a stable and accurate synchronization is the key requirement of gPTP.

Although the CANHUB board integrates a hardware-timestamping PHY, which provides very stable and low-jitter timestamp measurements, it also offers only a limited correction step for adjusting its local clock. In practice, the maximum adjustment that the CANHUB can apply at each update is not sufficient to compensate for the natural frequency difference between the two nodes, which in our measurements is approximately 25 ppm. If the CANHUB were used as the Slave, this limitation would prevent it from continuously tracking the Grand Master's clock, causing the synchronization offset to drift over time.

The White Board, instead, provides a wider clock-adjustment range and allows finer compensation of the frequency error. For this reason, it is more suitable to act as the gPTP Slave, where continuous and precise clock corrections are required. The CANHUB, despite its reduced correction capability, is the the candidate for the Grand Master role.

4.4.1 Reference Planes for Timestamping

Another limitation comes from the fact that the S32K344 MCU integrates only one Ethernet MAC interface. The EMAC, using an RGMII connection, is directly linked to the SJA1105QS switch through port 0. Because of this architecture, the timestamp reference plane is shifted further away from the actual physical point where the frame enters the PHY. In practice, the switch cannot be bypassed, and every gPTP message must pass through it before being forwarded to port 1, which is the one connected to the MR-CANHUBK344 board.

- EMAC (on WB board)
- SJA1105 switch (port 0)
- TJA1101 PHY
- 100BASE-T1 cable
- TJA1103 PHY on CANHUB board

However, this reference-plane shift does not introduce a significant mismatch in the delay measurement between the two nodes. Both ingress and egress timestamps are taken at a consistent reference level on each side of the link. On the MR-CANHUB board, the reference plane is located directly at the PHY, which supports hardware timestamping. On the WB board, instead, the reference plane is placed at the EMAC level 4.4.1, where the RTD firmware is responsible for filtering and identifying PTP messages that must be timestamped.

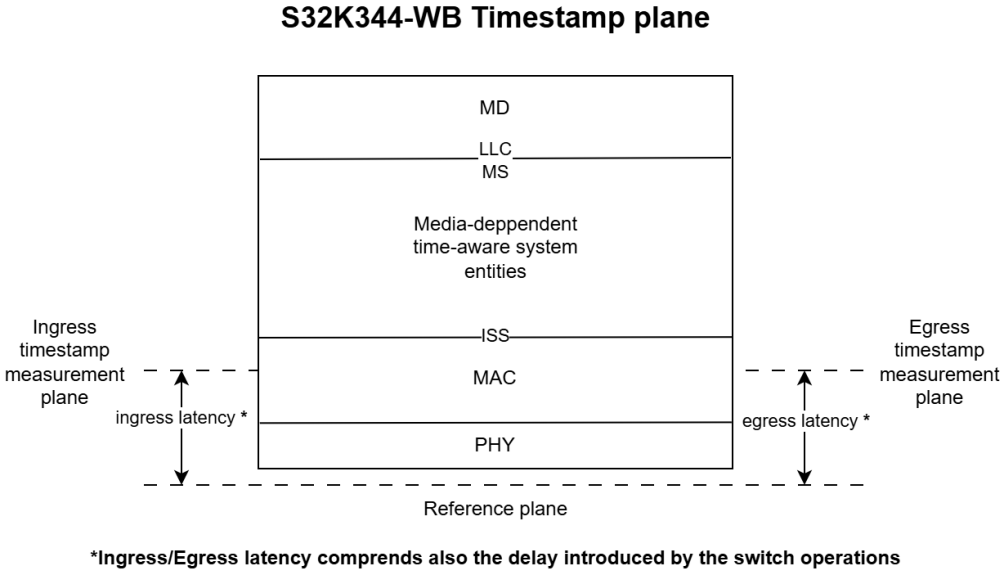

Figure 4.6: Timestamp and reference plane difference in the WB board

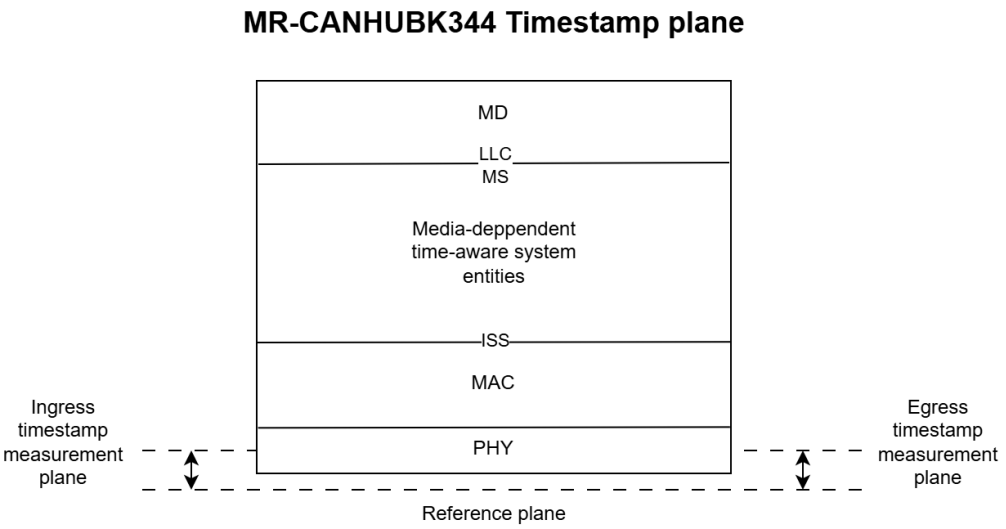

Figure 4.7: Timestamp and reference plane difference in the CANHUB board. Note that ingress/egress delay is little comparing to 4.6 because of the presence of TJA1103, capable of HW timestamping

4.5 Software Implementation

The software development initially started using S32 Design Studio 4 together with the SDK4. However, during the early integration phase it became clear that this environment did not allow a proper integration of the gPTP stak, since it relies on the configuration system and driver introduced only in S32DS 5 and RTD 5.0.0. For this reason the whole project was migrated to the newer toolchain, based on the RTD(Real-Time Drivers) that are compliant with AUTOSAR 4.7, providing a more suitable foundation for integrating the gPTP synchronization modules.
The core component in this implementation is the EMAC module of the S32K devices, and in particular the GMAC driver delivered by NXP. This driver is responsible for managing Ethernet frame reception and transmission, handling DMA buffer descriptors, and retrieving the timestamps required bu the gPTP protocol.

4.5.1 Ring Buffers

The GMAC RTD driver is responsible for managing both the reception and transmission of Ethernet frames through the EMAC. It relies on the DMA engine to automatically fill and empty two ring buffers: one for incoming frames (RX ring) and one for outgoing frames (TX ring). Each ring is implemented as a FIFO structure, ensuring that all frames are handled in the same order in which they arrive.

Frame reception is performed using the *Gmac_Ip_ReadFrame()* function, which allows the application to check whether new frames are present in any of the RX ring slots. When the DMA writes a frame into a buffer, the software retrieves it, processes it, and then returns the buffer to the ring for future use.

The TX side follows a similar structure, with an additional mechanism to track the status of sent frames. The GMAC driver stores transmission-related metadata—such as the EMAC egress timestamp—inside dedicated descriptor fields. These descriptors remain occupied until the function *Gmac_Ip_GetTransmitStatus()* is called. To correctly retrieve the status of a specific frame, the function must be provided with the corresponding data pointer and frame length.

To manage this association between the original frame and its descriptor, a small software structure has been implemented:

- **Data**: pointer to the frame payload;
- **Length**: length of the transmitted frame;
- **ring**: index of the descriptor used in the TX ring;
- **inUse**: flag indicating whether the slot is still waiting for status retrieval.

```
typedef struct {
    uint8 *Data;
    uint8 Length;
    uint8 ring;
    bool inUse;
} DescrBuffer;
```

This structure ensures that the software always knows which frames have already been transmitted, allowing correct extraction of timestamp information and freeing TX descriptors in a safe and controlled way for future transmissions.

Given the limited on-chip memory and the fixed structure of the GMAC descriptor rings, the choice of both the slot size and the number of entries for each buffer is a critical design decision. An undersized RX ring may lead to buffer saturation and frame loss, especially during high traffic loads. This aspect becomes even more important because both RX handling and TX-buffer cleanup are performed through periodic polling. For this reason, the ring configuration must balance memory usage, expected traffic load, and the required real-time responsiveness of the application.

4.5.2 Main Loop Architecture

The application does not rely on an external real-time scheduler (e.g., FreeRTOS). This choice is motivated by two main factors: (i) the overall system complexity is low, and (ii) the required determinism for handling gPTP messages demands precise and predictable execution timing. For these reasons, all real-time functionalities are executed inside a single, statically scheduled infinite loop.

The main loop consists of time-triggered functions, each executed with a specific period. This approach guarantees tight control over the timing of message processing, timestamp handling, and gPTP state-machine execution.

The execution pattern of the main loop is the following:

- **Every 50 µs:** `Eth_Poll()` This is the highest-frequency task, responsible for all Ethernet-related polling. It internally performs:
  - `rx_check()` Checks whether the RX ring contains a newly received frame. It identifies whether the incoming frame is VLAN-tagged and, if it contains a gPTP message, triggers `EthIf_RxIndication()`, which performs message decoding and type classification.
  - `tx_free_buffer()` Iterates over all TX buffer slots to verify whether transmitted frames can be released, ensuring the availability of free descriptors and preventing queue buildup.
- **Every 2 ms:** `Eth_PollLinkStatus()` Periodically checks the PHY link status. This ensures correct link operation and contributes to maintaining stable synchronization in the gPTP state machine.

- **Every 10 ms:** `GPTP_TimerPeriodic()` Updates the internal gPTP timer and drives the execution of the gPTP state machine. This function manages the scheduling and timing of essential protocol events such as Sync generation, Pdelay message exchange, and timeout handling.

4.5.3 Polling Frequency and Buffer Dimensioning

The initial stack configuration recommended calling `Eth_Poll()` every 2–3 ms. However, under realistic traffic conditions this period proved insufficient: even moderate background traffic combined with periodic gPTP messages caused the RX ring to fill faster than it could be drained. For this reason, the polling interval was reduced to 50 µs to guarantee timely descriptor consumption and avoid RX queue saturation.

To correctly size the RX and TX rings, the worst-case throughput was estimated by considering all periodic protocol traffic and the potential background load circulating in the network. The theoretical maximum bandwidth of a 100BASE-T1 link is:

$$R_{\max} = 100 \text{ Mbps} = 12.5 \text{ MB/s}$$

If `Eth_Poll()` is executed every $T_p = 50$ µs, the amount of data that can arrive between two polling events is:

$$B_{\text{window}} = R_{\max} \cdot T_p = 12.5 \cdot 10^6 \, \frac{\text{B}}{\text{s}} \cdot 50 \cdot 10^{-6} \, \text{s} = 625 \text{ B}$$

This value represents the maximum amount of data that the DMA may insert into the RX ring between two polling cycles. Considering that the system must also handle:

- - Sync frames every 125 ms (Layer 2, approx. 100–110 B including headers),
- - Pdelay Request / Response / Response Follow-Up (approx. 100–120 B per message),
- - possible application or background traffic (potentially several Mbit/s),

the ring must be dimensioned to absorb short-term traffic bursts and jitter in task execution.

To evaluate the required size of the RX and TX descriptor rings, it is useful to estimate how different combinations of payload size and transmission frequency contribute to the overall network load. Since the GMAC DMA assigns one descriptor per frame regardless of its actual payload length, the key parameter is therefore the *frame arrival rate* rather than the total number of bytes per unit time.

The theoretical maximum throughput of 100BASE-T1 is 100 Mb/s. For a periodic transmission of Ethernet frames of payload size L_{payload}, sent every T microseconds, the generated load is:

$$R = \frac{(L_{\text{payload}} + L_{\text{overhead}}) \cdot 8}{T},$$

where L_{overhead} includes Ethernet header, VLAN tag (if present), preamble, SFD and IFG. In the test environment this overhead is approximately 38 B.

Some representative examples are:

- **512 B payload every 100 µs:**

$$R = \frac{(512 + 38) \cdot 8}{100 \cdot 10^{-6}} \approx 44.1 \text{ Mb/s}$$

This traffic level is already sufficient to keep the RX ring constantly active and can easily generate short bursts where multiple descriptors are filled before the next polling cycle.

- **512 B payload every 50 µs:**

$$R \approx 88.3 \text{ Mb/s}$$

This configuration pushes the link close to saturation and quickly exposes any lack of buffering or insufficient polling frequency.

- **1400 B payload every 200 µs:**

$$R = \frac{(1400 + 38) \cdot 8}{200 \cdot 10^{-6}} \approx 57.5 \text{ Mb/s}$$

Even though the average throughput is moderate, the large frame size increases the risk that multiple descriptors are consumed back-to-back when the source sends two or three large frames in rapid succession.

These estimations must also account for the fixed traffic generated by the gPTP protocol. Each Sync/Follow-Up pair (125 ms period) and each Pdelay Request/Response/Response-Follow-Up cycle contributes a small but non-negligible number of frames that cannot be postponed or dropped without affecting clock stability. In burst conditions, these protocol frames compete with application traffic for RX descriptors and further justify the use of larger descriptor pools.

Based on these considerations and on empirical stress tests, the final configuration assigns 15 descriptors of 1536 B to the RX ring on the slave node and 10 descriptors of 1024 B to its TX ring. The master similarly uses 15 TX descriptors of maximum size and 10 small RX descriptors. This setup proved robust under realistic worst-case conditions, ensuring that the polling interval of 50 µs is sufficient to avoid ring saturation even during short high-intensity bursts.

4.5.4 gPTP Stack Integration

The stack implementation chosen for this implementation is the **gPTP S32K3xx V0.9.0** and in order to better understand the internal behavior of the driver and to work directly with the low-level RTD drivers, I decided to integrate the stack without relying on the AUTOSAR Ethernet modules. Using the AUTOSAR layer (ETH_43 and the whole MCAL abstraction) would have hidden most of the mechanisms that I needed to study, such as how frames are parsed, how timestamps are captured, and how the GMAC DMA interacts with the descriptor rings. Moreover, adopting an AUTOSAR workflow would have required configuring the entire project through the TRESOS tool, which was not necessary for the scope of this thesis.

For these reasons, I replaced all the ETH_43 function calls with their corresponding low-level RTD driver functions. This was possible thanks to some reference projects and example applications that showed how the GMAC driver is actually used underneath the AUTOSAR interface. By following this approach, the integration remained as transparent as possible, giving me full visibility over the hardware behavior and complete control over the timing-critical operations.

This design choice also made it possible to handle, in a clean and consistent way, the two different timestamp-collection methods used in the project:

- the CANHUB board timestamps at the **PHY boundary**, providing the closest reference to the physical medium;

- the White Board timestamps at the **MAC boundary**, further from the physical link due to the mandatory switch traversal;

Managing these two paths at the AUTOSAR layer would have been significantly more complex, while the RTD-level integration allowed a straightforward access to the PHY registers on the CANHUB side and to the DMA-generated `RxInfo`/`TxInfo` structures on the White Board.

4.5.5 Switch Configuration and Integration

In this project, part of the validation campaign was performed using an SJA1105Q/S automotive Ethernet switch.

The switch configuration focuses on enabling accurate forwarding of gPTP messages while avoiding any feature that could mask the natural behavior of the clock-synchronization process. For this reason, all Time-Sensitive Networking (TSN) scheduling functionality (e.g. time-aware shaper, enhanced CBS, scheduled traffic) has been intentionally left disabled. Only the mechanisms strictly required to guarantee full frame transparency and timestamp propagation were activated.

Table 4.3: Summary of SJA1105Q/S Switch Configuration for gPTP Validation

Feature / Table	**Configuration**	**Description / Motivation**
Device mode	100BASE-T1, 5 ports	Full-duplex links to PHYs and MCU
Host port selection	Port 0	Source/sink of PTP frames generated by the MCU
MAC filtering	PTP multicast enabled	Ensures timestamping of event messages (Sync/Pdelay)
Address learning	Dynamic learning enabled	Simplifies network setup and MAC allocation
Forwarding behavior	All ports reachable except ingress	Full path availability for gPTP traffic
Queue configuration	Queue 2 and 7 enabled	Reduced latency for synchronization frames
VLAN support	VLAN 1/2	Separation of gPTP test traffic when needed
Interface mode	RMII/MII according to HW design	Matches MCU and PHY electrical connectivity
Broadcast domain	Limited per ingress port	Prevents unnecessary flooding

Table 4.3 summarizes the most relevant configuration aspects for the synchronization tests.

This configuration does not aim to minimize residence time at all costs, but rather to ensure that gPTP traffic is treated with the appropriate priority compared to best-effort frames. In particular, frames carrying synchronization messages are mapped to higher priority queues, so that they are forwarded earlier than lower-priority traffic whenever contention occurs on an egress port.

As a result, the measurements reflect the behavior of a realistic automotive Ethernet network, where time-critical control traffic is prioritized over background load, but is still subject to queueing effects when the link is heavily utilized. The focus of the validation is

therefore on how the slave node maintains synchronization in the presence of congestion, not on an artificially ideal, unloaded network.

CHAPTER 5

Experimental Methodology and Evaluation Framework

This chapter presents the methodology adopted to assess the synchronization performance of the implemented gPTP stack on the S32K344 master–slave setup. Instead of enumerating individual tests, the discussion is organised by *evaluation metrics* and *phenomena of interest*, each of which is analysed under different traffic and configuration conditions (baseline, VLAN-enabled operation, variable load, variable frame size, and transient events). Each section briefly describes what is evaluated, why it matters, how the measurement is obtained, and how it should be interpreted.

5.1 Data Acquisition and Logging Architecture

In order to analyze the performance limits of the network and of the gPTP implementation, a set of metrics is periodically collected on the Slave node and transmitted using a custom UDP logging message.

The experimental setup consists of the CANHUB board, configured as Grand Master for the reasons described in Section 4.4, connected to the WB board (acting as Slave) through a 2.40 meter UTP cable. The WB board provides a dedicated diagnostic Ethernet port on which all logging messages are transmitted. A PC running Wireshark is connected to this port using a standard Cat6 RJ45 cable and captures all the UDP log packets for offline analysis.

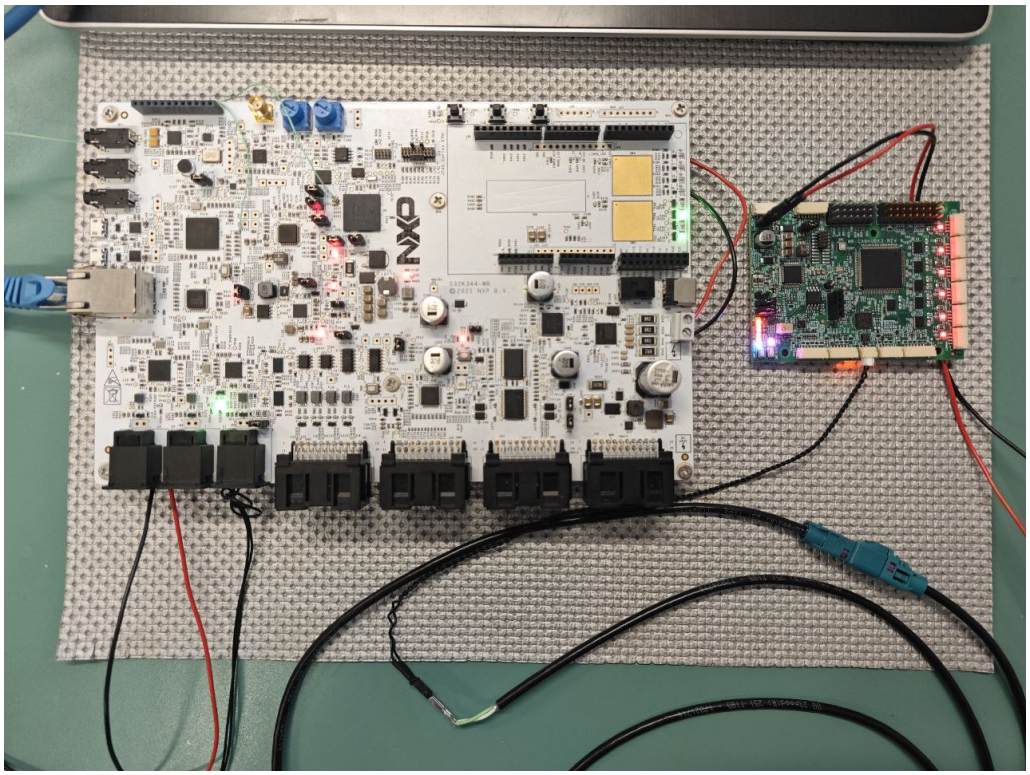

Figure 5.1: Experimetal setup for data validation.

Data analysis is performed by extracting several key parameters from the Slave node, which must synchronize its local clock to the Grand Master. The following fields are included in each log entry:

- **T1, T2, T3, T4:** timestamps associated with the Peer Delay (Pdelay) message exchange. Their meaning follows the definition given in Section **??**. T1 and T4 are taken on the Slave, whereas T2 and T3 refer to timestamps obtained from the Grand Master.
- **offset_from_master:** the estimated clock offset between Slave and Grand Master, computed using the origin timestamp provided in the Follow-Up message.
- **meanPathDelay:** the measured mean propagation delay of the link, derived from the Pdelay exchange.
- **rateRatio:** the ratio between the Slave's clock frequency and the Grand Master's clock frequency (i.e., the frequency drift). This corresponds to the `f64RateRatio` value produced by the Sync state machine.
- **syncIntervalIndex:** the time interval between two consecutive Sync messages, expressed in nanoseconds and measured locally on the Slave.
- **sequence_id:** the incremental sequence identifier embedded in Sync and Pdelay messages, enabling detection of lost messages on the receiving side.
- **local_timestamp:** the timestamp taken by the WB board at the moment the log packet is transmitted. It provides a local temporal reference for aligning log entries.

Each logging message contains ten consecutive entries of the structure shown below. The log packet is sent only when the internal counter `log_index` reaches 10. This counter

is incremented every time a Pdelay Response, Follow-Up, or Psync message is received. In other words, for every 10 messages received between Pdelay and Psync events, a single log packet is transmitted containing the state of the stack after the last 10 messages. This batching strategy reduces the transmission overhead on the network and takes advantage of the relatively large MTU allowed by standard Ethernet frames, enabling a single log packet of 760 bytes.

To avoid unnecessary memory expansion, all padding policies of the S32 ARM-based MCU are explicitly disabled, ensuring that all fields are tightly packed without alignment padding.

```
#pragma pack(push, 1)
typedef struct __attribute__((packed)) log_gptp {
    gptp_def_timestamp_sig_t T1, T2, T3, T4;
    uint32 offset_from_master;
    uint32 meanPathDelay;
    float64_t rateRatio;
    uint32 syncIntervalIndex;
    uint16 sequence_id;
    Gmac_Ip_TimestampType local_timestamp;
};
#pragma pack(pop)
```

Notes on fields:

- T2 and T3 are not arrival times on the Slave, but values coming from or measured on the Grand Master during the Pdelay procedure.
- `offset_from_master` is calculated based on the Follow-Up message, following the gPTP algorithm.
- `syncIntervalIndex` represents the actual time elapsed between two consecutive Sync messages in nanoseconds.
- `rateRatio` corresponds to the Sync state machine output, not the Pdelay neighbor rate ratio.

5.2 Clock Offset Analysis

5.2.1 Offset stability

In order to evaluate the quality of the gPTP software integration, an initial baseline test was performed by measuring the clock offset between the Grand Master and the Slave over a continuous 15-minute run. The goal of this analysis is to obtain a clean reference behavior of the system, without additional traffic or VLAN prioritization, and to extract key metrics such as mean offset, standard deviation, and peak deviation values. In this test, raw gPTP frames are forwarded through the switch without QoS, allowing us to characterize the intrinsic synchronization capability of the network.

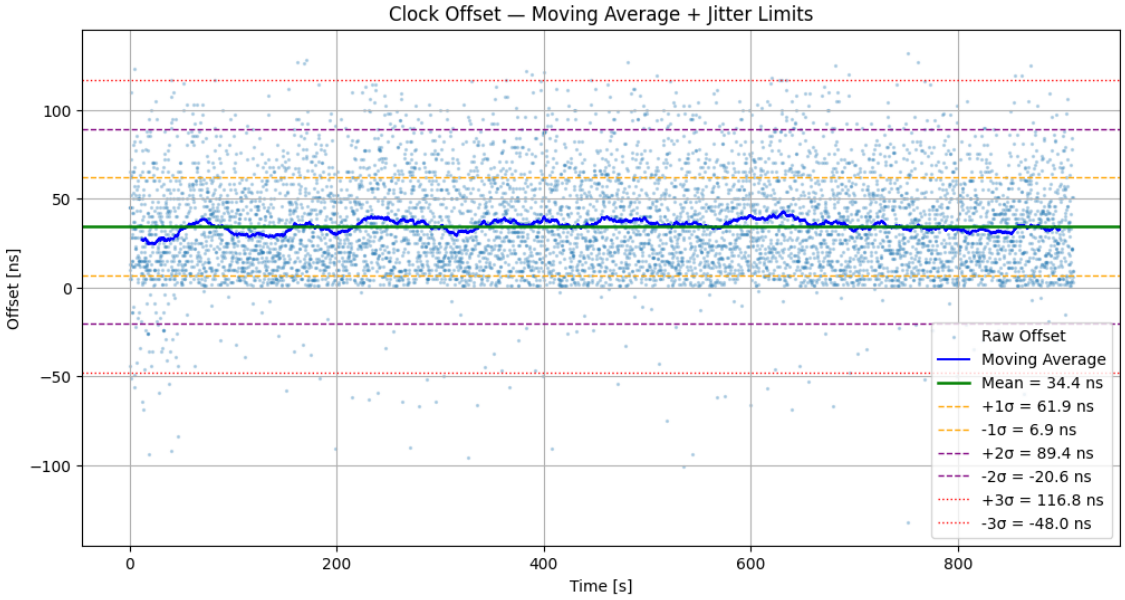

Figure 5.2: Offset baseline over a 15-minute run

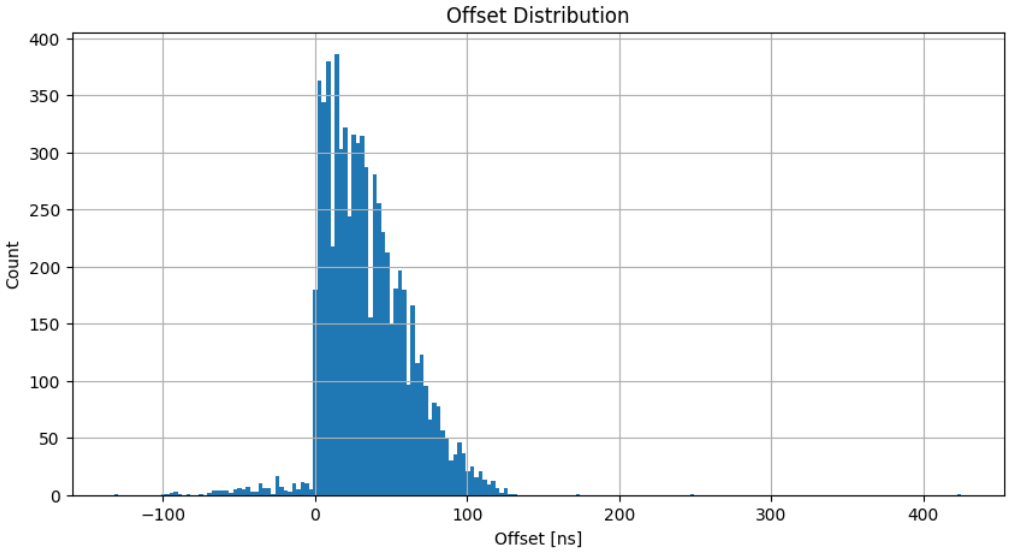

Figure 5.3: Statistical distribution of the computed offsets

Figure 5.2 shows that the offset naturally fluctuates around a mean value of **34.40 ns**, with a standard deviation of **27.48 ns**. Most samples remain within the $\pm 1\sigma$ band, indicating a stable clock synchronization performance. More than 95% of the measurements fall inside the $\pm 2\sigma$ interval (approximately +89 ns and −20 ns), confirming that the jitter is mostly limited to expected variations due to packet forwarding through the SJA1105 switch. This values are also in the range precision required by one of the most used stack such as AVnu [20], that set acceptable deviation in a range between ±80 ns from the mean value.

The histogram in Figure 5.3 highlights that the offset distribution is strongly concentrated between ≈ 0 ns and 50 ns, where the Slave clock is tightly aligned to the Grand Master. A few outliers are present, with a minimum value of **-132 ns** and a maximum of **+132 ns**. These rare spikes correspond to occasional queuing delays inside the switch when gPTP event messages are forced to wait behind ongoing Ethernet frames. This be-

havior is consistent with the sending of LOG message to the loggin port that has to be forwarded by the switch.

5.2.2 Path Delay analisys

The second parameter analyzed in this baseline scenario is the mean path delay, computed by the gPTP Peer Delay mechanism. Figure 5.4 reports the measured propagation delay over a 15-minute run, after filtering the invalid start-up samples (i.e. delay equal to zero).

The average path delay measured by the Slave is approximately:

$$Pdelay = 15505 \text{ ns}$$

with a fluctuation of about ±20 ns during the entire measurement.

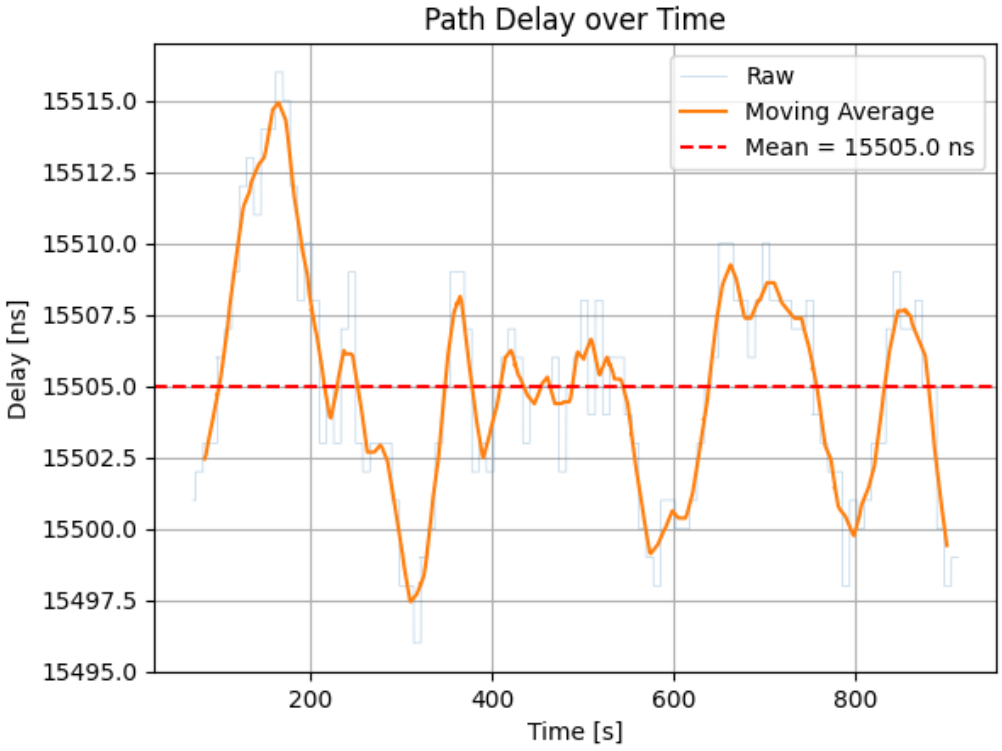

Figure 5.4: Pdelay mean estimation

This value is consistent with the expected architecture-dependent delay. Although only two meters of 100BASE-T1 cable are used (whose physical propagation delay would be below 10 ns), the timestamp reference plane on the WB board is located at the EMAC output. Therefore, the measured value also includes several internal latency components: (i) the forwarding pipeline of the SJA1105 switch, (ii) the PHY ingress/egress delays on both nodes, and (iii) the internal MAC-to-PHY interface latency.

As a result, instead of observing a delay centered around a few nanoseconds, the measured mean path delay stabilizes around 15.5 µs. This suggests that the switch residence time dominates the overall propagation measurement.

To validate this interpretation, an empirical measurement was performed using a Picoscope oscilloscope. The TX_EN signal from the S32K344 GMAC and the TX_EN signal at the output of the SJA1105 port connected to the TJA1101 PHY were captured simultaneously. The observed time difference between these two reference points confirmed a switch residence latency of approximately 14.9 µs, which aligns with the MeanPathDelay estimated by the gPTP Peer Delay mechanism.

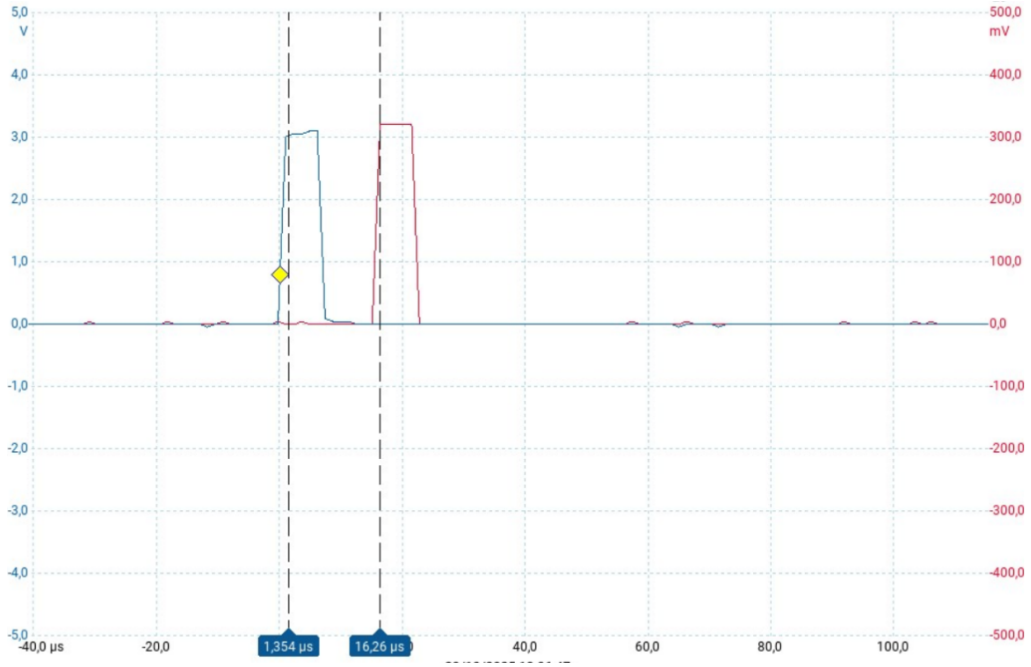

Figure 5.5: Switch residence time: green signal is the Tx enable signal from the MCU and red one is the *TX_EN1* signal from the SJA1105 on the port1

The periodic oscillation visible in 5.4 is attributed to normal buffering effects inside the switch and the dynamic gPTP frequency correction process. Importantly, the stability demonstrated in this test confirms that the Peer Delay mechanism operates correctly and that no long-term drift is observed under low-traffic conditions.

5.3 Impact of Network Traffic on Synchronization Performance

After a baseline estimation of the synchronization without any traffic, this section investigates how network traffic load and frame size influence the quality of clock synchronization between the gPTP master and the slave node. The goal is to identify the limits of the system under different network conditions, and to quantify the effect of additional UDP traffic, variable payload sizes, and VLAN configuration on the slave clock offset.

5.3.1 Network Load Evaluation

To evaluate the robustness of the gPTP synchronization under realistic traffic conditions, the Grand Master node was configured to generate IPv4 dummy traffic toward the Slave. These dummy frames contain a 32-bit incremental counter in the payload, allowing the receiver to easily detect packet losses and relate them to synchronization degradation.

The objective of this campaign is to investigate how increasing network load impacts:

- the clock offset stability between Master and Slave,
- the number of lost packets under congestion,
- network components behavior under stress.

Several traffic profiles were tested by combining different payload sizes and sending intervals, as summarized in Table 5.1. The combinations that reach or exceed the nominal 100 Mbps bandwidth of 100BASE-T1 are highlighted.

Payload	On-wire bytes	Bits/packet	1 ms	200 µs	100 µs	50 µs
128 B	198	1 584	1.58	7.92	15.84	31.68
512 B	582	4 656	4.66	23.28	46.56	**93.12**
1400 B	1470	11 760	11.76	**58.80**	**117.60**	**235.20**

Table 5.1: Traffic profiles and theoretical bandwidth usage (Mbps).

The test execution followed a cyclic structure: every 120 s the traffic condition was changed while the synchronization remained active. For each payload size, the following sequence was applied:

1. No dummy traffic (reference baseline)
2. Dummy traffic @ 1 ms period (120 s)
3. No dummy traffic (stabilization)
4. Dummy traffic @ 200 µs period (120 s)
5. No dummy traffic (stabilization)
6. Dummy traffic @ 100 µs period (120 s) (except 1400 B, to prevent exceeding line rate)
7. No dummy traffic (stabilization)
8. Dummy traffic @ 50 µs period (120 s) (except 1400 B, same motivation)
9. No dummy traffic (reference closing window)

This full sequence was executed under three different network configurations:

- **No VLAN**: all traffic in best-effort forwarding
- **Single VLAN**: gPTP traffic prioritized over dummy frames
- **Dual VLAN**: separation of control and data traffic with different priorities

The following sections present the impact of each traffic condition and VLAN configuration on synchronization stability and data integrity.

5.3.2 Test Under Load Without VLAN Prioritization

The first stress scenario was executed without any VLAN tagging or QoS prioritization, meaning that all network traffic was treated as best-effort. In this configuration, the gPTP synchronization messages (Sync, Follow-Up, Pdelay $_{Req/Resp}$) compete directly with dummy high–bandwidth UDP traffic injected from the Grand Master.

Figure 5.6 shows the clock offset evolution on the Slave over the experiment duration. The system initially operates with stable synchronization, maintaining offsets below 100ns. However, once the dummy traffic load increases, several synchronization messages fail to reach the Slave within their expected periodic deadline of 125ms. This loss creates long gaps in synchronization updates, resulting in abrupt jumps in the recovered time.

When synchronization packets are dropped, the rate-ratio compensation alone is insufficient to maintain alignment, and the offset grows quickly into the microsecond range. Peaks up to 50 are observed, followed by a recovery phase lasting approximately 5s during which the state machine gradually re-aligns the local time to the master reference.

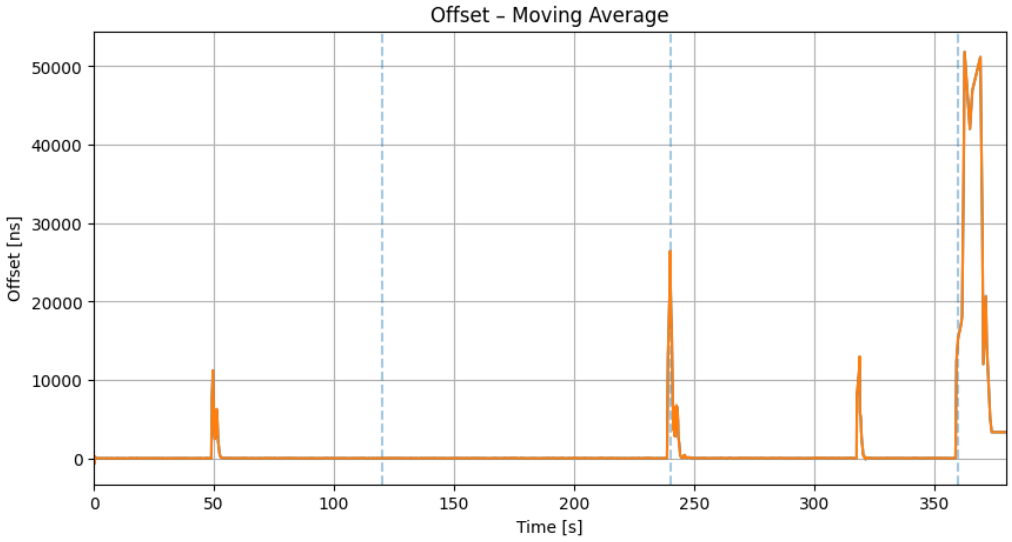

Figure 5.6: Clock offset evolution without VLAN or traffic prioritization

The corresponding mean propagation delay estimation is reported in Figure 5.7. A first transient phase is visible in the first 40s, where the Peer Delay mechanism accumulates enough valid measurements to initialise its filters. After convergence, the delay remains mostly stable around 15500ns, but a visible step change occurs during the congestion period, indicating that timestamp asymmetries introduced by switch buffering also perturb the computed link delay.

Once the traffic condition returns to normal, the delay estimation progressively reconverges to the nominal value.

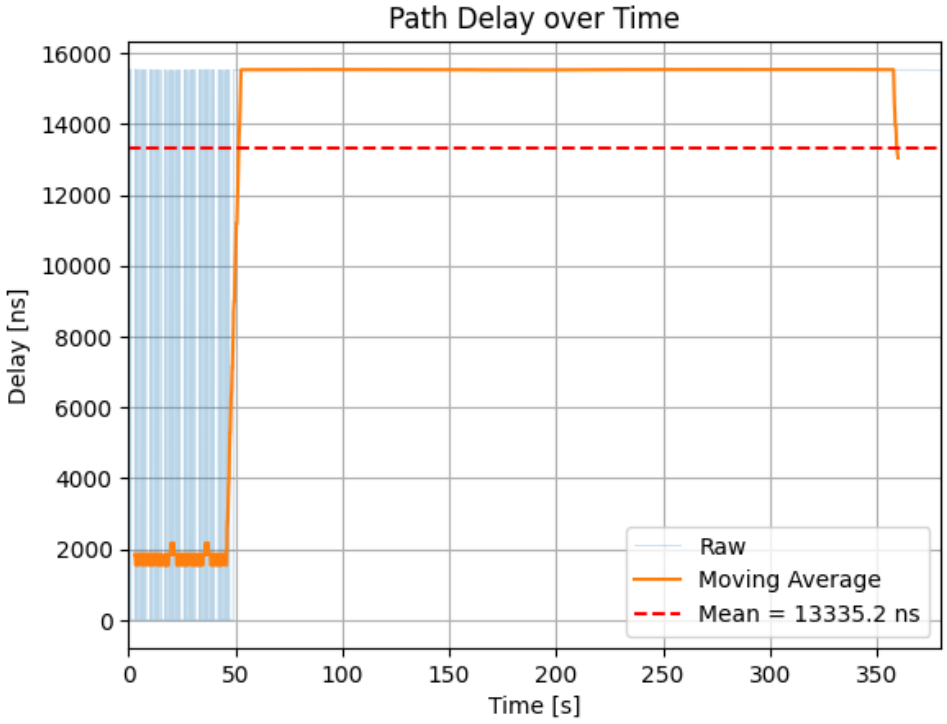

Figure 5.7: Peer delay estimation under best-effort network load

These results clearly demonstrate that:

- Without prioritization, dummy traffic preempts time–critical PTP traffic at the switch output queues.
- Loss of consecutive Sync/Fup messages leads to large offset excursions, up to tens of microseconds.

This configuration is therefore not acceptable for deterministic automotive synchronization, confirming the necessity of TSN queueing and VLAN prioritization in real deployments.

5.3.3 VLAN-based Traffic Isolation under Load

In the second group of experiments, the same burst pattern described in Section 5.3.1 was executed with VLAN configuration enabled on the SJA1105QS switch. The goal is to understand how traffic prioritisation and logical separation help to preserve clock synchronisation when the link is stressed with increasing dummy load.

Two VLAN configurations were evaluated:

- **Case A – Single VLAN**: all traffic is tagged on VLAN 1; gPTP frames use the highest priority, while dummy frames use a lower one.
- **Case B – Dual VLAN**: gPTP frames are tagged on VLAN 1, dummy traffic on VLAN 2; priorities remain the same as in Case A.

Table 5.2 summarises the configuration used in both runs.

Table 5.2: VLAN and PCP configuration for gPTP and dummy traffic

Case	VLAN topology	gPTP PCP	Dummy PCP
A	VLAN 1: gPTP + dummy	7	2
B	VLAN 1: gPTP, VLAN 2: dummy	7	2

In both cases, gPTP traffic is mapped to the highest priority (PCP = 7), while dummy packets are transmitted as best-effort (PCP = 2). The burst sequence is the same used for the baseline tests: every 120 s a new traffic pattern is enabled, and the transition instants are marked by vertical dashed lines in the plots.

Offset behaviour with single and dual VLAN

Figure 5.8 and Figure 5.9 show the clock offset between Master and Slave when VLAN-based prioritisation is enabled. The plotted curve is a moving average (window of 50 samples) of the raw offset values, used to improve readability while still preserving the main trends.

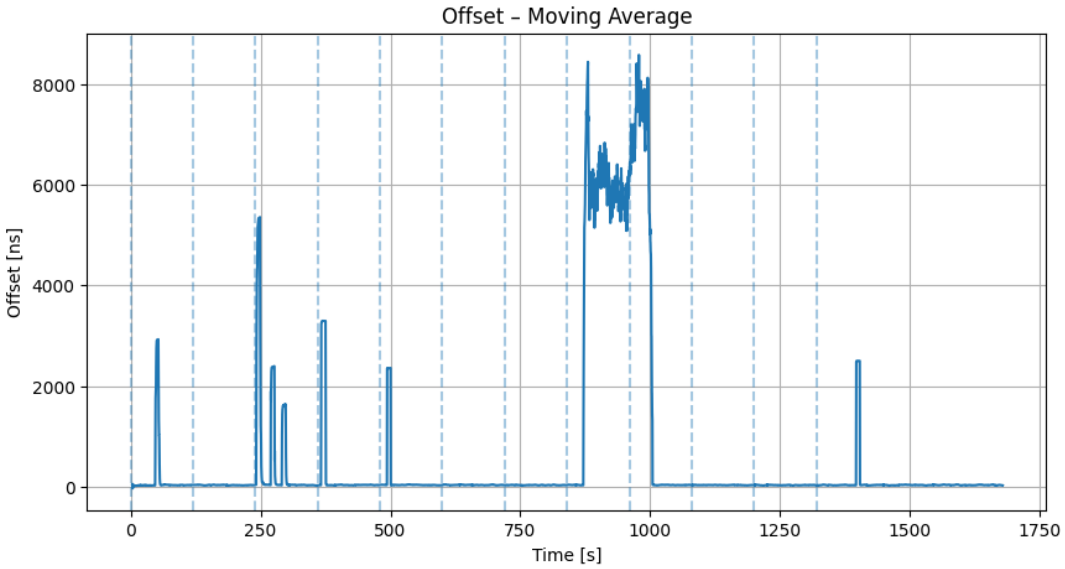

Figure 5.8: Offset moving average – single VLAN (Case A). Dashed lines mark the beginning of each 120 s burst.

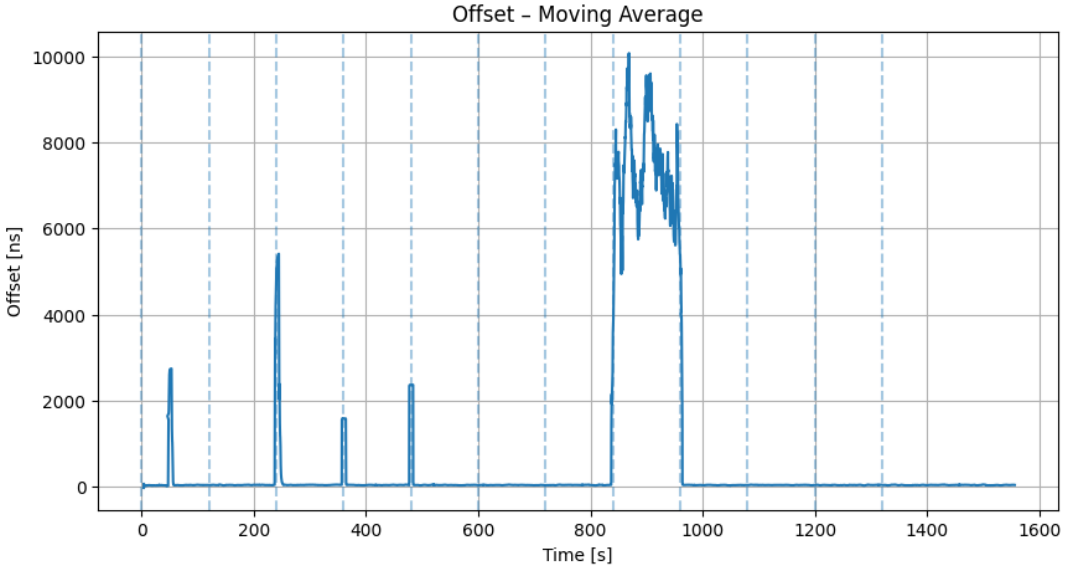

Figure 5.9: Offset moving average – dual VLAN (Case B). Dashed lines mark the beginning of each 120 s burst.

At the beginning of the run, a short transient is visible around $t \approx 50$ s, where the offset shows a first spike that is not associated to any high-load burst yet. This behaviour is also present in the no-VLAN baseline and is mainly related to the convergence of the gPTP state machine and the initial adjustment of the Slave clock. After this phase, the system stabilises and the offset returns into the same range observed in the baseline test without dummy traffic (typically between 20 ns and 60 ns).

For the bursts with moderate load (e.g. payloads up to 512 B at periods of 1 ms and 200 μs), both configurations behave in a very similar way. In correspondence with each dashed line, the moving average exhibits small spikes, but these excursions last only a few seconds, while each burst window has a duration of 120 s. This indicates that the switch and the gPTP stack are able to absorb the temporary increase of contention, lose at most

a few Sync/Follow-Up messages, and quickly re-align the Slave to the Grand Master.

A significantly different behaviour appears when the burst with payload 1400 B and period 200 μs is enabled (around $t \approx 840$ s). This configuration corresponds to an offered load of approximately

$$R \approx 11.76 \text{ Mbps} \times 5 \approx 58.8 \text{ Mbps},$$

that is, about 60% of the nominal link capacity. In this phase, both Case A and Case B show a strong degradation of the offset: the moving average rises up to several microseconds, and the raw data contains peaks of up to about 27.3 μs. The time series clearly shows that:

- the offset increase is *persistent* over the whole 120 s burst;
- once the burst ends, the offset converges again to the original steady-state band (tens of nanoseconds).

In other words, the very aggressive burst (1400 B @ 200 μs) periodically pushes the system close to its limits, causing short intervals where too many Sync and Follow-Up frames are delayed or discarded. However, as soon as the number of valid time messages becomes sufficient again, the gPTP algorithm is able to recover and re-synchronise the Slave clock.

From a qualitative point of view, no major difference is observed between the single-VLAN and dual-VLAN configurations: having gPTP and dummy traffic on separate VLAN IDs does not completely eliminate the problem, because both flows still share the same physical egress queues inside the switch. What makes the real difference is the *priority* (PCP = 7 vs PCP = 2), not the VLAN ID itself.

Correlation with packet loss

To better understand the origin of the offset spikes, an additional metric was extracted from the log: for each received Sync/Follow-Up message, the sequence ID was compared with the previous one. Whenever two consecutive sequence numbers were not adjacent, a counter of "missing packets" was incremented. This metric is then plotted together with the offset in Figure 5.10 and Figure 5.11.

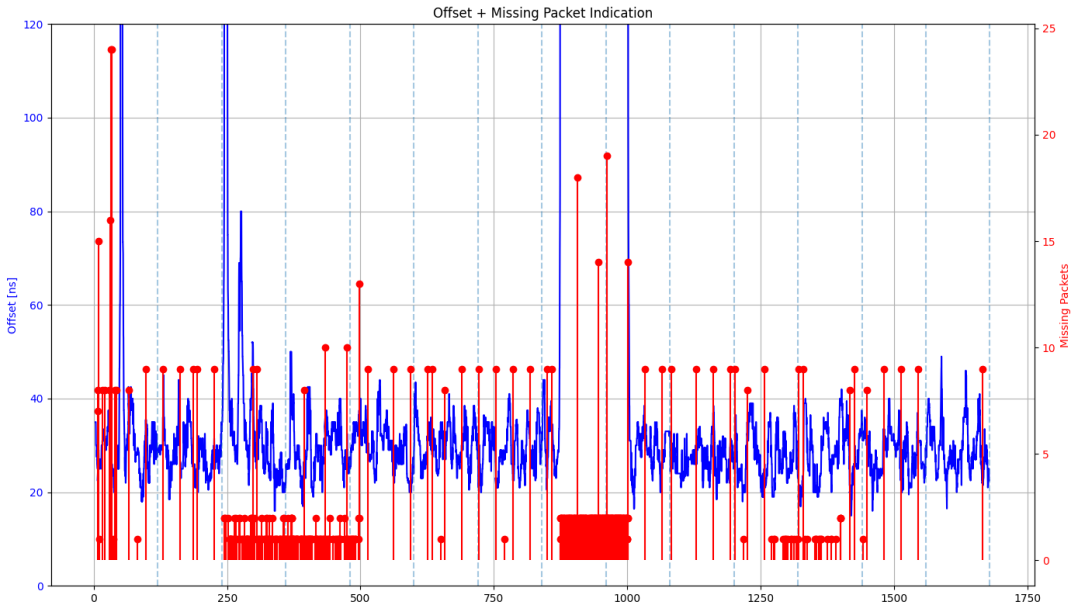

Figure 5.10: Offset and missing Sync/Follow-Up packets – single VLAN (Case A). Blue: offset; red: estimated number of lost packets.

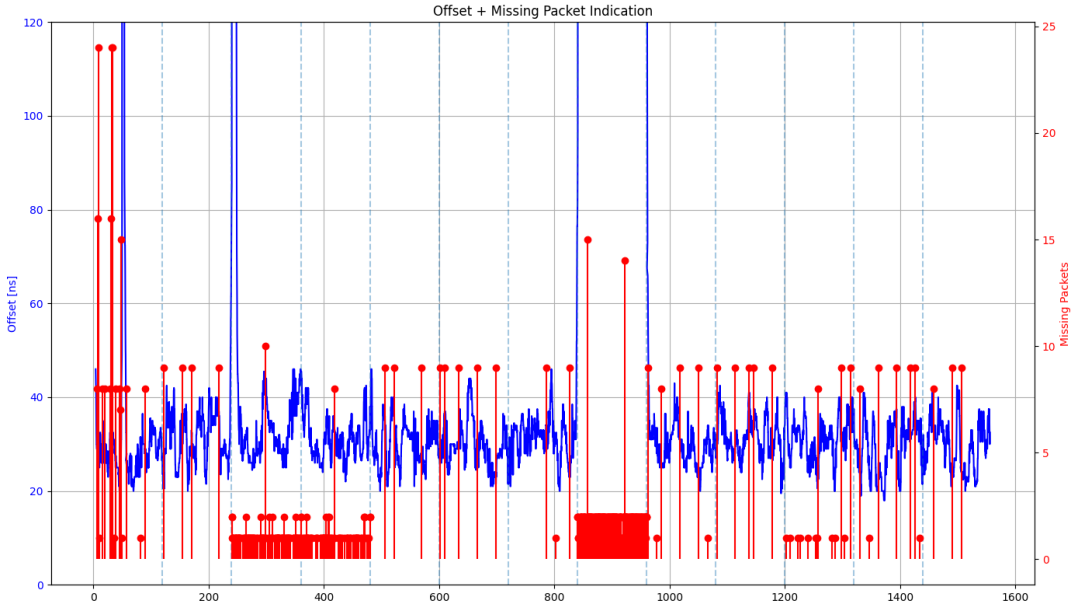

Figure 5.11: Offset and missing Sync/Follow-Up packets – dual VLAN (Case B).

The plots show a clear correlation: the largest offset excursions are always aligned with bursts of missing packets. The highest concentration of losses occurs:

- during the first large spike around $t \approx 240$ s, where the system is still stabilising under load;
- during the 1400 B @ 200 μs burst, where the number of missing packets becomes much higher.

Outside these intervals, small groups of missing packets are still present along the whole run. These isolated events are most likely due to occasional delays in message

processing on the Slave, temporary filling of the egress queues in the switch, or minor disturbances introduced by the logging mechanism itself. In these cases, the effect on the offset is limited and short-lived.

The two VLAN configurations exhibit a very similar loss pattern: separating gPTP and dummy frames on different VLAN IDs slightly improves isolation but does not avoid losses when the offered load becomes too aggressive.

Path Delay stability

Finally, Figures 5.12 and 5.13 report the behaviour of the *meanPathDelay* estimated by the Pdelay mechanism during the same runs.

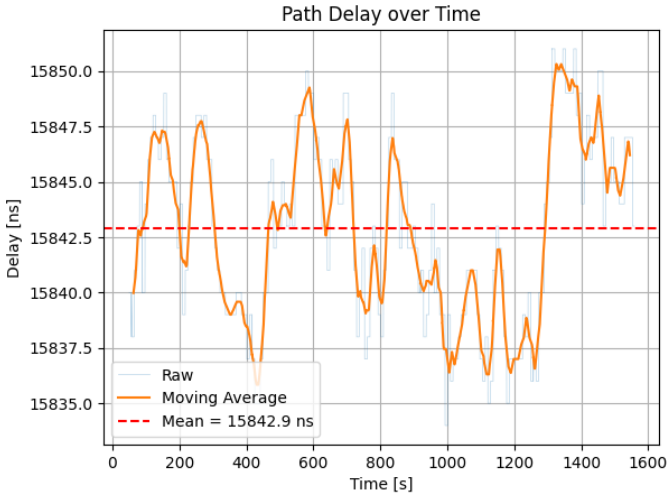

Figure 5.12: Path Delay over time with moving average – single VLAN (Case A).

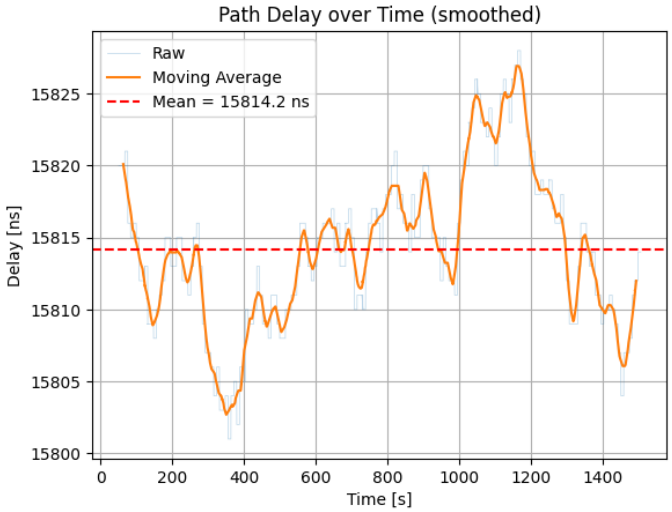

Figure 5.13: Path Delay over time with moving average – dual VLAN (Case B).

After a short start-up phase (the first tens of seconds), where the averaging filter is still filling its window, the estimated path delay stabilises around:

$$\text{meanPathDelay} \approx 15.8\ \mu\text{s},$$

with relatively small fluctuations (a few tens of nanoseconds) over the entire duration of the experiments.

This behaviour is consistent with the hardware architecture explained in Section 4.4.1. The measured delay does not include only the pure cable contribution (a 2 m 100BASE-T1 link would introduce less than 10 ns), but also the EMAC-to-switch path on the WB board, the internal residence time of the SJA1105QS pipeline, and the PHY ingress/egress delays on both sides. Because of the chosen reference planes, the Pdelay value is dominated by the switch residence time, which was empirically measured around 14.9 μs using an oscilloscope on the TX enable signals.

The reason why the Path Delay remains so stable, even when the offset shows large spikes, lies in the symmetry of the Pdelay computation. Neglecting noise, the mean path delay is estimated as:

$$d_{\text{mean}} \approx \frac{(T_4 - T_1) - (T_3 - T_2)}{2},$$

where T_1 and T_2 are the sending and receiving timestamps for the request, and T_3 and T_4 are the sending and receiving timestamps for the response. Any additional latency introduced by the switch or by the PHYs affects both the forward and reverse direction in a very similar way.

This explains why, even during the most critical burst (1400 B @ 200 μs), the offset temporarily degrades while the meanPathDelay remains essentially constant.

Synchronization recovery after congestion

To better quantify the resilience of the synchronisation mechanism, the recovery phase was examined after the end of the most critical burst (1400 B @ 200 μs). Figures 5.14 and 5.15 show the evolution of the offset immediately after the dummy traffic stops.

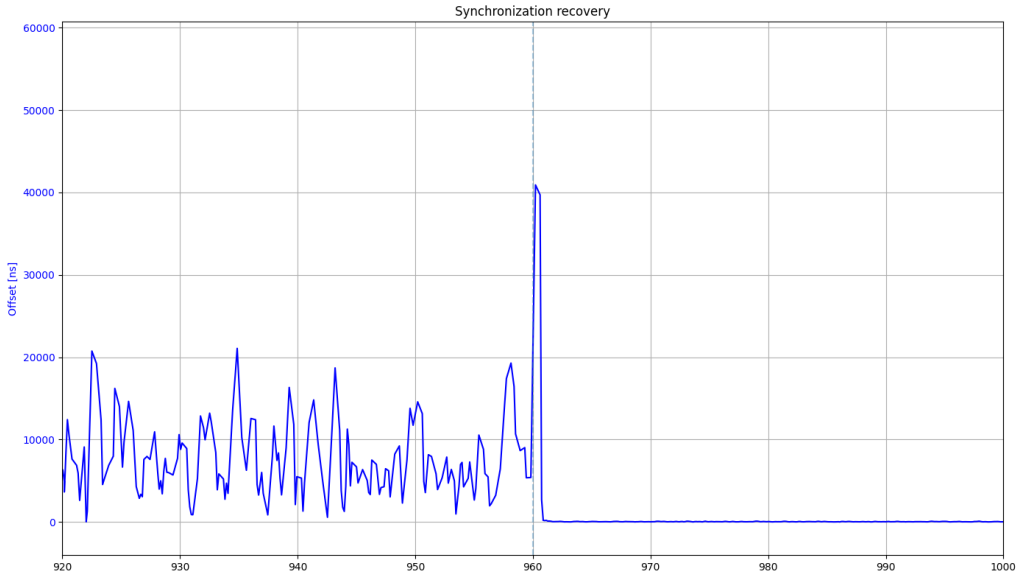

Figure 5.14: Offset recovery after the high-load burst. Dashed line marks the instant where the dummy traffic returns to idle load.

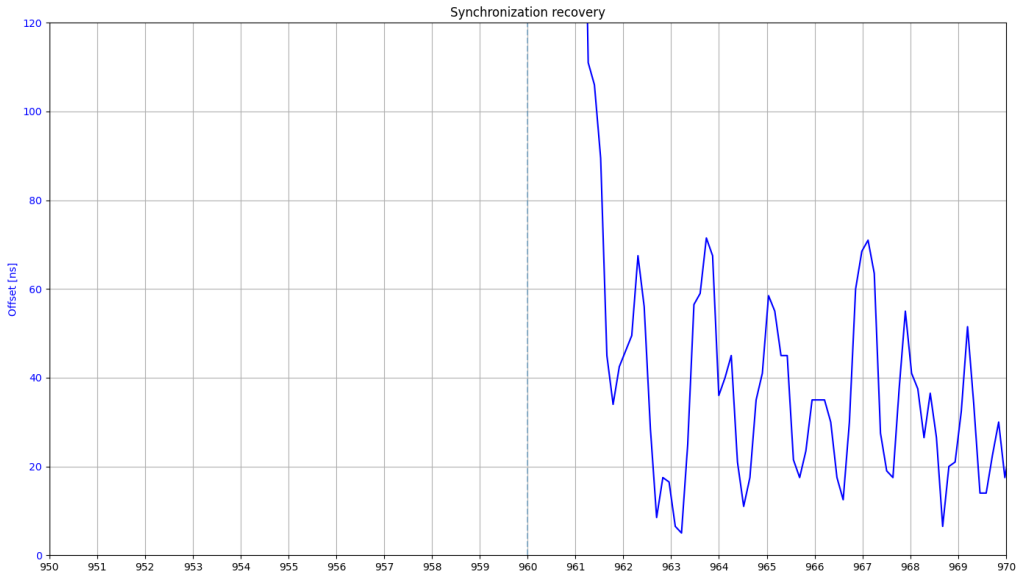

Figure 5.15: Zoom of the recovery region, showing return to the steady-state band within a few seconds.

When the bandwidth is free again (dashed line), the Slave clock begins to compensate the accumulated frequency error and reduces the phase offset. Despite the large excursion during the burst (over 40 µs in the worst case), the system restores a stable synchronisation within approximately 2–3 s, where the offset falls again inside the nominal steady-state band (roughly between 20 ns and 60 ns).

These results demonstrate that the synchronisation mechanism remains stable and robust even after short periods of severe congestion. Although the offset can temporarily degrade due to queuing delays and packet loss, the system always recovers autonomously without requiring a re-initialisation of the protocol state machines or a forced re-selection of the Grand Master.

Summarized the observations

- VLAN-based prioritisation (PCP = 7 for gPTP, PCP = 2 for dummy) is *effective* in keeping the offset within tens of nanoseconds for all moderate traffic patterns.
- Offset spikes are short compared to the 120 s duration of each burst: even when a new traffic pattern starts, the synchronisation is recovered in a few seconds.
- Both VLAN configurations (single and dual) behave similarly; separating gPTP and dummy traffic on different VLAN IDs does not radically change the behaviour, because they still share the same physical queues inside the switch.
- The most critical condition is the 1400 B @ 200 μs burst (around 60% link utilisation), where the system experiences bursts of packet loss and a temporary increase of the offset up to several microseconds.
- Despite these disturbances, the gPTP state machine always manages to re-align the Slave once a sufficient number of Sync/Follow-Up messages is correctly received again.

Overall, these experiments show that VLAN-based prioritisation significantly improves robustness compared to the no-VLAN case, but very aggressive traffic patterns can still

degrade the synchronisation due to loss or excessive delay of gPTP messages. This behavior will be further discussed in the concluding remarks of this chapter.

CHAPTER 6

Conclusions and future work

The goal of this thesis was to investigate the BroadR-Reach communication protocol, its position within the ISO/OSI stack, and its practical usability in automotive networking. The document introduced the fundamental characteristics of automotive Ethernet and the 100BASE-T1 physical layer, with a specific focus on the implementation and validation of a small network prototype synchronized through the gPTP protocol. This synchronization mechanism represents a key enabler for Time-Sensitive Networking (TSN) and Audio/Video Bridging (AVB), which are increasingly required in modern automotive systems.

The experimental results, obtained from a physical setup rather than simulation, highlight the importance of the hardware timestamp reference plane: accurate synchronization is only possible when timestamps are captured as close as possible to the physical network interface. Solutions based on MAC-level timestamps are strongly affected by static latencies (such as switch residence time and PHY delays), whereas timestamping inside the PHY (e.g., TJA1103) allows significantly improved alignment to the true network timing.

Furthermore, the study demonstrated that network management plays a fundamental role: the use of VLANs and Quality of Service allows high-priority gPTP traffic to remain stable even in presence of heavy best-effort load, maintaining offset fluctuations within ± 80 ns under normal conditions. Despite the implementation being influenced by the latency of the SJA1105 switch and the configuration of transmit/receive ring buffers, the network successfully met synchronization expectations for automotive environments, confirming the validity of the design.

Future Work

The proposed implementation represents a solid starting point for extending the setup towards more realistic topologies. Future work may include promoting the SJA1105 switch to a *PTP relay instance* and introducing a third node (e.g., an additional CANHUBK344 board), allowing the evaluation of multi-hop synchronization behavior and cumulative latency effects. Further improvements should also target advanced queue scheduling and switch buffer configuration, to enhance performance under persistent network congestion. Also new feature could be developed to extend the stack functionality for example by implementing the BCMA mechanism to find the best master in case the of the current one is failing off.

Outlook on Automotive Networking Evolution

The automotive industry is undergoing a major architectural transition: moving from traditional domain-based networks built on CAN, LIN and FlexRay towards centralized,

zonal architectures. In this new paradigm, automotive Ethernet is becoming the backbone technology due to its scalability, deterministic communication capability, and support for high-bandwidth data streams.

Standards such as 100BASE-T1 and 1000BASE-T1 will play a crucial role in connecting sensors, cameras, radars, and high-performance computing units required for ADAS and autonomous driving. In parallel, TSN extensions will guarantee strict timing behavior for safety-critical applications.

The solution analyzed in this thesis therefore not only serves as an academic prototype, but also aligns with current and future industrial trends, confirming that Ethernet-based synchronization is a key technological component in the evolution of automotive networks.

CHAPTER 7

Acknowledgements

Desidero esprimere la mia più sincera gratitudine alla società TRAMA Engineering per avermi ospitato durante lo svolgimento di questa tesi. Un ringraziamento speciale va a Flavio Cerruti, per avermi proposto un progetto che si è rivelato una vera e propria sfida di ricerca: partire dai fondamenti per arrivare ad una soluzione concreta, utile e altamente formativa, sia dal punto di vista tecnico che personale.

Ringrazio inoltre Pierluigi Allio per la sua disponibilità e per aver condiviso con me parte del suo straordinario bagaglio di esperienza, accompagnandomi con costanza e professionalità in questi mesi di lavoro.

Un ringraziamento va alla mia famiglia e ai miei amici, che mi hanno sostenuto durante tutto il mio percorso al Politecnico, fatto di impegno e continue sfide. Grazie per aver creduto in me anche nei momenti più difficili.

Infine a Laura: grazie per avermi supportato e, soprattutto, sopportato in questo cammino; grazie per la forza e la fiducia che mi trasmetti ogni giorno, semplicemente grazie per essere chi sei.

Bibliography

[1] M. van den Beld, "How zonal e/e architectures with ethernet are enabling software-defined vehicles," 2023.

[2] C. Ronan, "Network topology." `https://lightningxvpn.com/blog/en/network-topology/`.

[3] LAN/MAN Standards Comitee, *IEEE Standard for Ehernet*, 2022. [Online; Accessed 2025].

[4] Open Alliance, *BroadR-Reach® Physical Layer Transceiver Specification For Automotive Applications*, 2014. [Online; Accessed 2025].

[5] D. Porter, "100base-t1 ethernet: the evolution of automotive networking," 2018.

[6] ISO, *Road vehicles — Diagnostic communication over Internet Protocol (DoIP) — Part 2: Transport protocol and network layer services*, 2019. [Online].

[7] *FC602 - USB 100BASE-T1 Stick*. [Online].

[8] *MediaConverter 100/1000BASE-T1*. [Online].

[9] *RAD-Moon — 100BASE-T1 Media Converter*. [Online].

[10] *FL3X TAP 1000BASE-T1*. [Online].

[11] NXP, *S32K3XX — S32K3xx Data Sheet*. [Online Datasheet of S32K3xx family].

[12] G. L. Lo Bello, L.; Patti, "L. a. perspective on ethernet in automotive communications—current status and future trends," *Appl. Sci, 13, 1278,*, 2023. https://doi.org/10.3390/app13031278.

[13] Giovanni Popolizio, "Ethernet nel modello osi." `https://www.aiutocomputerhelp.it/protocollo-ethernet-facciamo-chiareza/`, 2021. [Online; Accessed 2025, 3 November].

[14] I. 802.3bw working group, *IEEE Standard for Ethernet Amendment 1: Physical Layer Specifications and Management Parameters for 100 Mb/s Operation over a Single Balanced Twisted Pair Cable (100BASE-T1)*, 2015. [Online].

[15] Teledyne, "Fundamentals of 100base-t1 ethernet." `https://www.teledynelecroy.com/doc/100base-t1-ethernet-appnote`.

[16] G. B. Correa, Simon, *Automotive Ethernet - The Definitive Guide*. 2022.

[17] L. S. Committee, *Frame Replication and Elimination for Reliability*.

[18] L. S. Committee, *Timing and Synchronization for Time-Sensitive Applications*.

[19] NXP, *TJA1103, ASIL B Compliant Automotive Ethernet 100BASE-T1 PHY Transceiver*. [Online 100BASE-T1 transceiver].

[20] AVnu, *802.1AS Recovered Clock Quality Testing*. [Online 100BASE-T1 transceiver].