POLITECNICO DI TORINO

Master Degree in Mechatronic Engineering

Master Degree Thesis

Ethernet Network in the Automotive field: Standards, possible approaches to Protocol Validation and Simulations

Supervisor
Prof. Claudio Ettore Casetti

Candidate
Carlo Cataldo

Internship Tutors
Intecs Solutions
Ing. Stefano Ferrario
Ing. Francesco Teti

Academic Year 2020 - 2021

Summary

The availability of advanced driver assistance systems (ADAS), the pressing demand for connectivity and increasingly complex and content-rich infotainment systems, have had, as first consequence, a considerable increase in data traffic and in the demand for solutions capable of ensuring high throughput and low latency. The intrinsic characteristics of the Ethernet Standard make it the ideal solution to the problem posed to automotive systems engineers, while, at the same time, citing advantages such as: reduction of complexity, weight and cost of wiring.
However, if on the one hand Ethernet Standard represents the solution to the problem of bandwidth availability, on the other it poses another important challenge: to define validation methodologies and strategies that allow to easily test the integrability of ECUs interconnected on Ethernet, as well as is currently the case for ECUs interconnected on CAN. In addition, simulator and test tools are not so obvious and well defined such as CAN networks.
Since Ethernet for local network cannot be used as it is for automotive purposes, due to different time requirements and Electromagnetic/Radio-Frequency Interferences, the objective is to deepen new and reusable standards from Physical Layer to Application Layer according to OSI model.
Testing methodologies will be also studied, in particular, regarding Conformance Testing will be referred to Open Alliance test specifications, instead for Performance Testing will be resumed RFC2544 and RFC2889 taken from Ethernet for local networks.
Finally, as a practical example, nodes connected on real-time Automotive Ethernet will be simulated, comparing the different time requirements, thanks to an open-source framework, called CoRE4INET, based on INET and OMNeT ++ simulator.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and motivations

This chapter describes the context of network architectures in the automotive field with its functional domains, the purpose of the thesis and consequently the reasons for using Automotive Ethernet.

1.1 Context

From the late 1800s until the 1970s the cars were entities purely mechanical or hydraulic. Since the 1970s, there has been an exponential increment of electronic parts inside a vehicle. Firstly, the electronic part played a small role, mostly for battery charging, starter, ignition and illumination. A few years later, the growing use of reliable and inexpensive electronic components and the widespread use of software replaced many mechanical parts. In fact, the first electronic control units (ECUs) were born with the main purpose of assisting the driver to control the vehicle, thanks to: antilock braking system (ABS), electric power steering (EPS), electronic stability program (ESP), engine control, lights control, doors control, windows control and many other functions.
Each new function in a vehicle is implemented with an ECU made of a microcontroller, many sensors and actuators. Nowadays the electronic part is dominating on the mechanical part, more than 70-100 ECUs, as shown in Figure 1.1, are embedded inside a vehicle. All of this to guarantee lower emission, longer range, higher safety and some comfort functions.[39]
Typically, the ECUs are classified into functional domains due to the different requirements and constraints: [2]

- **Powertrain**: real-time and safety control. The communication is for controlling the engine automatic transmission, hybrid control, transmission, gearbox, ...;
- **Chassis**: real-time and safety control. The communication is for controlling stability and dynamics, such as steering, braking, suspension, ...;

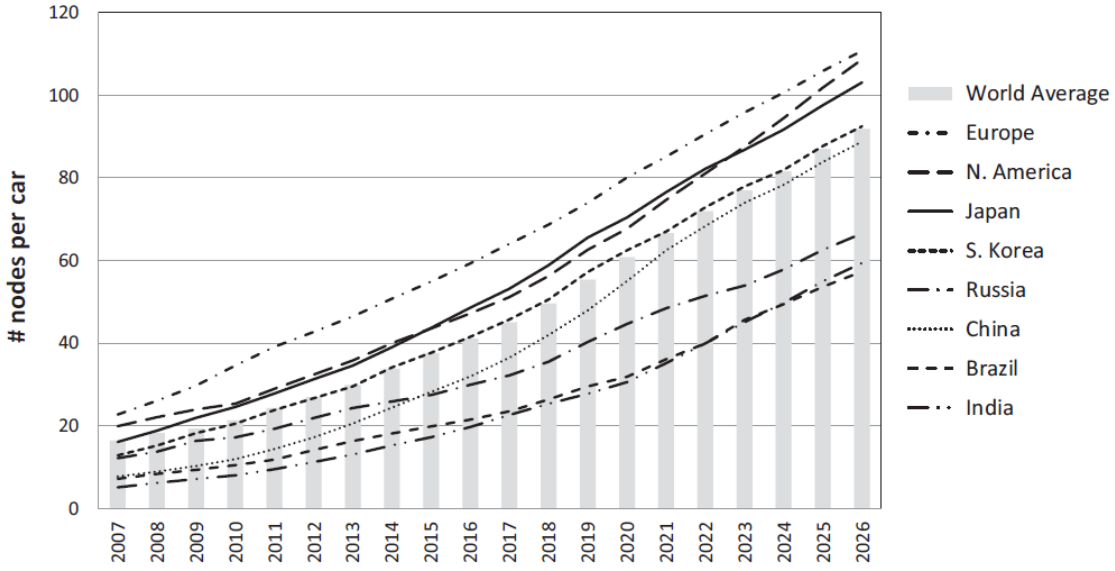

Figure 1.1. ECUs number on vehicle over time. Source: [59]

- **Body and comfort**: dashboard, climate control, doors, windows, lightning, mirrors, ...;
- **Driver assistance**: night vision, speed information, lane keeping assistance, parking assistance, ...;
- **Infotainment**: traffic information, car navigation, ...;
- **Entertainment**: car audio, video programs, phone calls, ...

Between the 1970s and 1990s the communication for connecting two ECUs was a point-to-point links. Given n, number of ECUs, n^2 channels are needed to connect them each other. Clearly, with a moderately high number of ECUs, this communication is unfeasible due to high cost, complexity and reliability.
For these reasons, since 1990s, a communication network, with a shared channel and consequently rules and protocols for accessing the bus have been used.[39]
The communications between two functional domains, described above, are not always the same. For some domain it can be important the time requirement with a real-time communication, instead for some other domain the first requirement is the bandwidth, as shown in Table 1.1. But there are also requirements in terms of: safety, efficiency of the error detection, redundant communication support and many others.
Consequently, all these communication requirements need various automotive network technologies such as: CAN, LIN, MOST, FlexRay, Automotive Ethernet like shown in Figure 1.2.

Domain	End-to-End Latency Requirements	Bandwidth Requirements
Powertrain	$< 10us$	Low
Chassis	$< 10us$	Low
Body and Comfort	$< 10ms$	Low
Driver Assistance and Driver Safety	$< 250us$ or $< 1ms$ depending on the system	20 - 100Mbps per camera
Human-Machine Interface	$< 10ms$	Varies by system, but high

Table 1.1. Domain time requirements

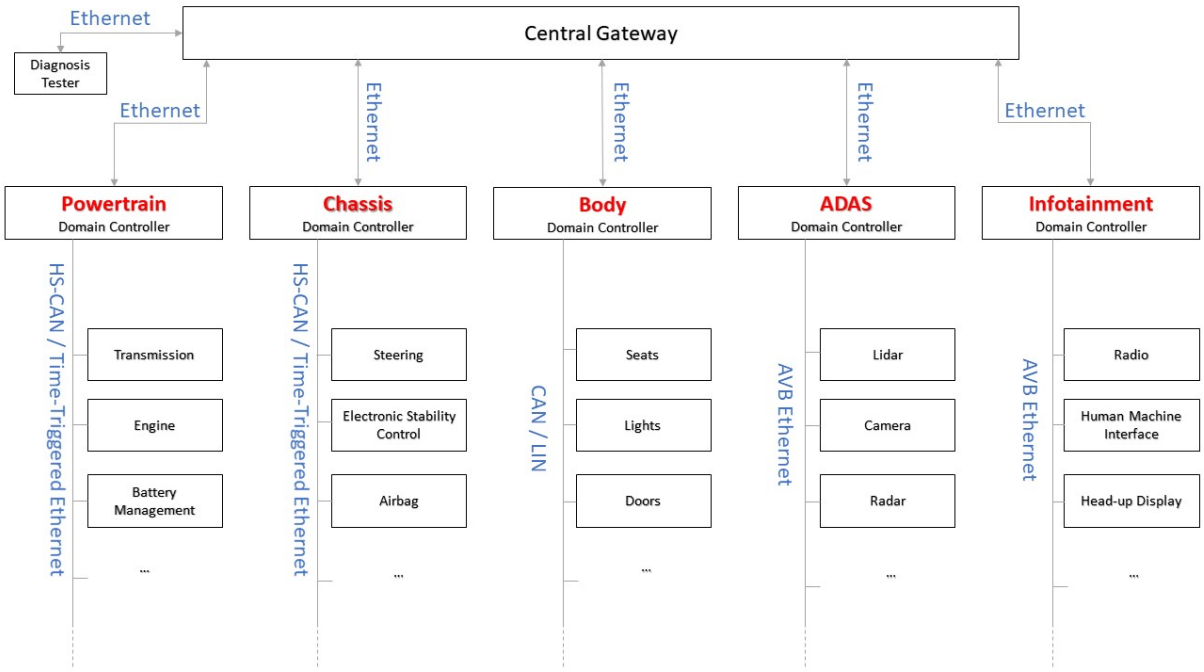

Figure 1.2. Domain centric architecture with different network technologies for each vehicle segment.

In particular, as it will deepen in the following chapters, Automotive Ethernet needs a hierarchical domain network architecture with Ethernet Switches like shown in Figure 1.3.
All these network technologies are not ideal, but each is proper for each different application, however they will be analysed in the next chapters, with particular emphasis on Automotive Ethernet.

1.2 Motivations and thesis scope

The large use of electronic components ECUs, actuators and sensors installed in vehicles, infotainment systems, driver assistance systems and so on have enormously increased the bandwidth usage for data transmission which cannot be satisfied by

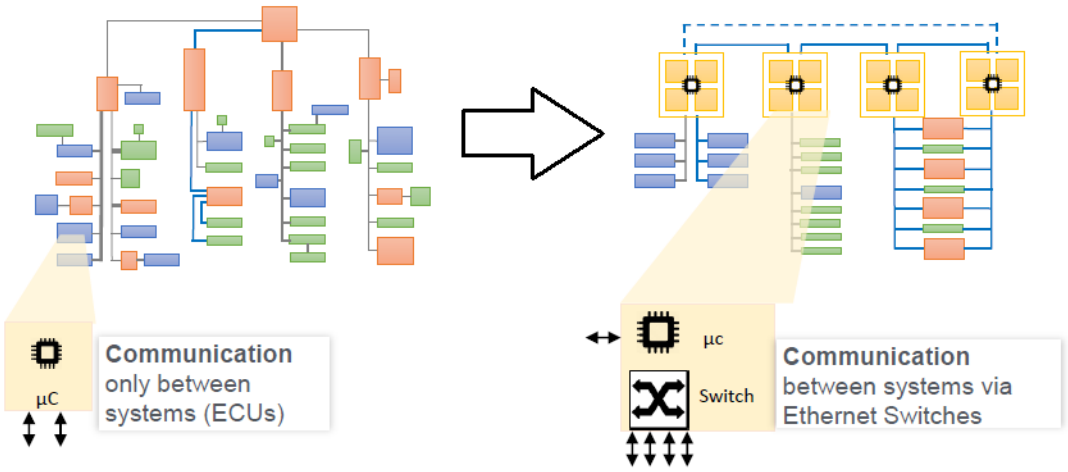

Figure 1.3. Domain distributed in a future network. Source: [24]

existing in-vehicle network, like CAN, FlexRay, MOST.
Ethernet deployment will satisfy the bandwidth problem and will reduce the cost in terms of electronics, cablings, network interfaces and onboard computing power. So, the objective is to guarantee high bandwidth requirements, small latency, good synchronization and network management requirements with a flexible and scalable network technology.
Unfortunately, Ethernet can not be used as it is, for example as in local networks, because: [27]

- It does not meet requirements on EMI (Electromagnetic Interference), RFI (Radio-Frequency Interference) and stringent temperatures;
- It can not synchronize the time between different devices;
- It can not be possible to transmit data shared by multiple types of sources (no control of bandwidth for different streams);
- It can not guarantee a pre-defined latency less than the order of microseconds.

Thanks to a worldwide partnership AUTOSAR (Automotive Open System Architecture), a non-profit alliance of automotive industry providers OPEN Alliance (One-pair-Ethernet Alliance) and many other research centers, the Ethernet standards have been adapted, integrated and modified for the automotive field.
However, Automotive Ethernet needs to be tested in order to avoid late discovery of problems, so conformance testing, interoperability testing and protocol validation are needed to validate the network technology.
The testing process has to be reliable and compatible with automotive V-Cycles,

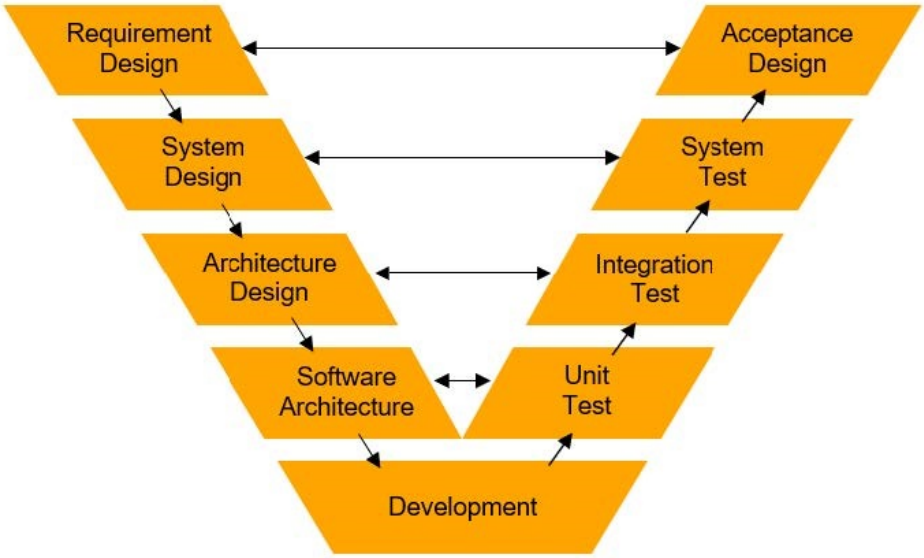

Figure 1.4. V-cycle software development process. Source: [3]

Figure 1.4, for all the development phase and for all components.
The goal of this thesis is to analyse and deepen the Ethernet network in automotive field and define methodologies and strategies to test and develop ECUs interconnected with Automotive Ethernet for all the seven layers of the Open Systems Interconnection model (OSI model), shown in Figure 1.5. Finally to understand how Automotive Ethernet stack and the related real-time protocols (i.e. AVB and TTEthernet) work, a simulation environment and an open-source framework will be studied.

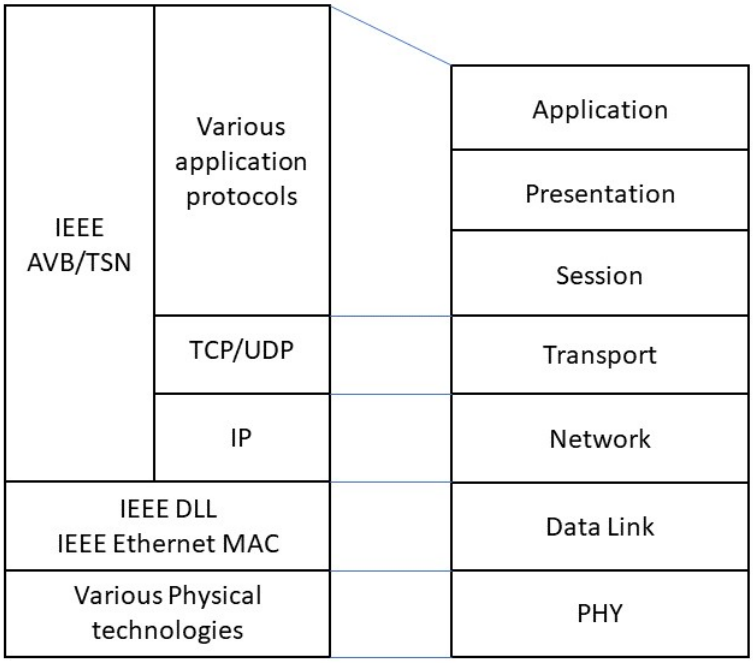

Figure 1.5. Open Systems Interconnection layers (OSI model), protocol stack of Automotive Ethernet.

1.3 Thesis outline

This thesis is organized in the following chapters:
Chapter 2 will describe, briefly, the usual automotive networks: CAN, LIN, MOST and FlexRay that are important to understand Automotive Ethernet.
Chapter 3 will introduce, firstly, Ethernet used in local networks, then the relevant subject to this thesis: Automotive Ethernet with differences and integrations each other. It will be explained the influence of Ethernet in automotive field, all the OSI model layers, from physical to application layer, in detail, with the main Application-related Protocols.
Chapter 4 will contain methodologies and strategies to test and validate conformances and performances for ECUs on Automotive Ethernet networks according to AUTOSAR, OPEN Alliance, Avnu Alliance and other research groups.
In *Chapter 5* will be presented an introduction to the simulation environments (OMNeT++, INET framework, CoRE4INET) and an example of Automotive Ethernet simulation with 3 nodes and 1 switch that communicate different types of traffic (AVB and TTE).
Finally, *Chapter 6* summarises the work with an outline of the results and proposes further works related to Automotive Ethernet, ECU testing and simulations.

Chapter 2

Conventional automotive networks

This chapter describes, briefly, the traditional automotive networks, such as: CAN, LIN, MOST and FlexRay that are important to understand Automotive Ethernet.

2.1 CAN

One of the first network technologies inside a vehicle was the Controller Area Network (CAN) developed by Robert Bosch GmbH in 1980. Today CAN bus is the most widely used network technology in vehicles.
Usually, on average, data rate of a CAN bus is between 125 Kbit/s and 500 Kbit/s, it depends how many nodes are connected. It is used, in particular, for powertrain, chassis and body domain for its robustness and bounded delays.
CAN is implemented as a shared bus with each node (see Figure 2.1), in fact two or more nodes can not communicate at the same time without a collision. Consequently, to avoid collisions, an arbitrage mechanism, Media Access Control (MAC), is needed. CAN uses Non-Return-to-Zero (NRZ) bit representation with a 5 bit as stuffing. [40]
The bus network is modelled like a logic AND with "1" recessive bit and "0" dominant bit. When all nodes, simultaneously, transmit a message with content 1, the bus will remain at 1. Instead, when one of the nodes transmits 0, the bus will be set to 0, allowing the node that sent 0 to continue sending the message, interrupting the transmission of all other nodes.
So the collisions are solved by a priority-based communication where lowest message identifier, contained in the header field of the frames, has highest priority.
CAN uses unshielded twisted pair (UTP) of copper wires where one cable is called CAN High, the other one CAN Low. The signal between two cables has a voltage of 2.5V; when a 0 dominant bit appears, CAN High voltage increases the nominal value by 1V, instead CAN Low voltage decreases, the nominal value by 1V; when a logic 1 it appears both CAN High and CAN Low have 2.5V. [4]

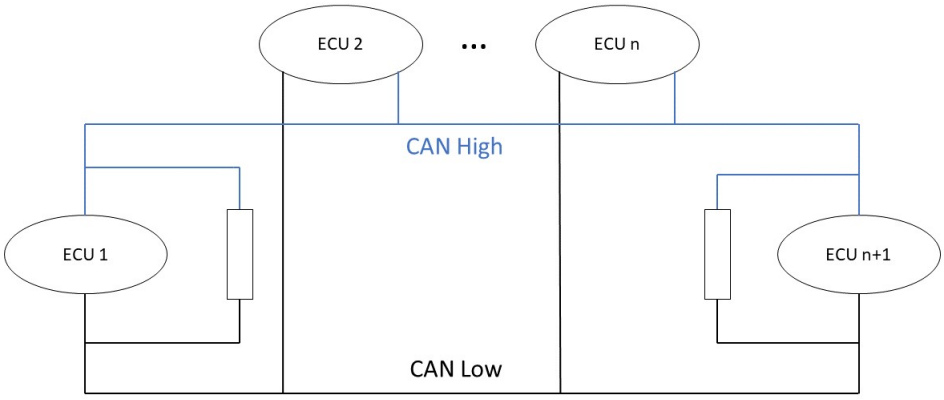

Figure 2.1. CAN bus.

A standard CAN data frame, which contains up to 8 bytes of data, is shown in Figure 2.2 where:

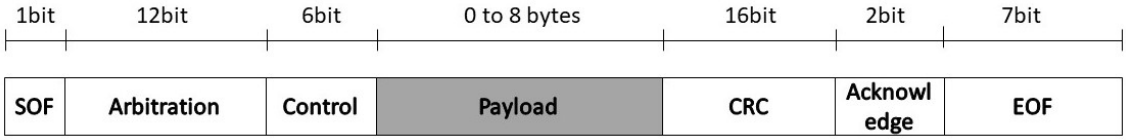

Figure 2.2. CAN data frame.

- **SOF**: Underline the start of frame transmission;
- **Arbitration**: first 11 bits are the Identifier ID which sets the priority of data frame, as explained before, lowest ID means highest priority. The 12th bit is the Remote Transmission Request equal to 0 for data frame;
- **Control**: this field contain the Payload length on four bits and more two bits for the protocol;
- **Payload**: Data;
- **CRC**: used to check the transmission, as explained shortly after;
- **Ack**: acknowledgment for CRC field;
- **EOF**: seven recessive bits to end the frame.

CAN also owns many error detections like:

- Comparison of CRC field between frame transmitted and received frame;
- Error counters (Transmit Error Counter TEC and Receive Error Counter REC) modified when a frame is correctly received or it contains an error.

When an error is detected by a CAN node, a particular error frame is sent to every other node connected to CAN network.[40]

2.1.1 CAN-FD

Particular attention is needed to CAN-FD, the last version of CAN network, with Flexible Data Rate. In this case the payload increases from 8 bytes of CAN to 64 bytes. This reduces overload and it increases efficiency.
CAN-FD is used for data rates from 2 to 5 Mbps, that data rate is reached because arbitration is not useful during some part of a frame. (See Figure 2.3).

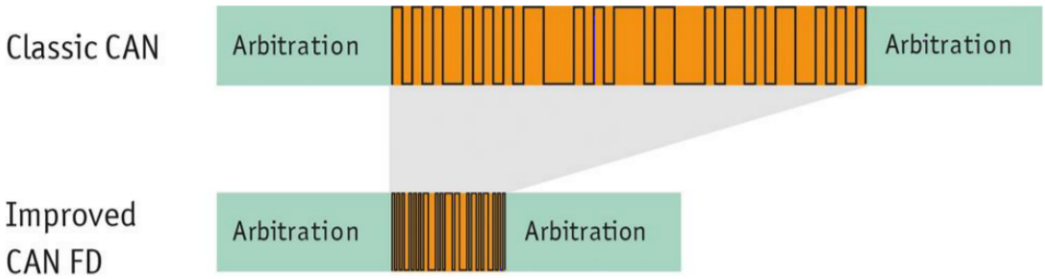

Figure 2.3. Arbitration field comparison between CAN and CAN-FD. Source: [66]

This last version shares the same functionalities of classic CAN. Instead, inside a frame, there is three new control bits: Extended Data Length (EDL), Bit Rate Switch (BRS), Error State Indicator (ESI). CRC field uses also more check bits to reduce the risk of undetected errors.
For completeness, there is a most recent CAN version, which is CAN XL with 10Mbps as data rate, but it has not yet been fully developed. [4] [35]

2.2 LIN

For some applications, such as windows, mirrors, seats, air conditioning control and many others, CAN technology is too expensive and provides too much protection for these applications that do not require safety and speed. So, in the late 1990s the LIN Consortium, a set of companies (Volkswagen, BMW, Volvo and Daimler), developed the Local Interconnect Network (LIN) with a first version released in November 2002. [40]
LIN was designed with 1-wire unshielded with a data rate limited to 20 Kbit/s, it

operates at nominally 12V with binary symbols: dominant 0 and recessive 1. [4] It is a simple Master/Slave bus system with two types of ECUs: a single LIN Master and many LIN Slaves, as shown in Figure 2.4. The Slave ECUs can transmit only after being queried by the Master ECU with a corresponding header. The scheduling is pre-defined in the design phase in a schedule table, which contains the frame list that have to be sent and the scheduled time period, in fact each frame is transmitted inside a time period, called Frame Slot. [40]

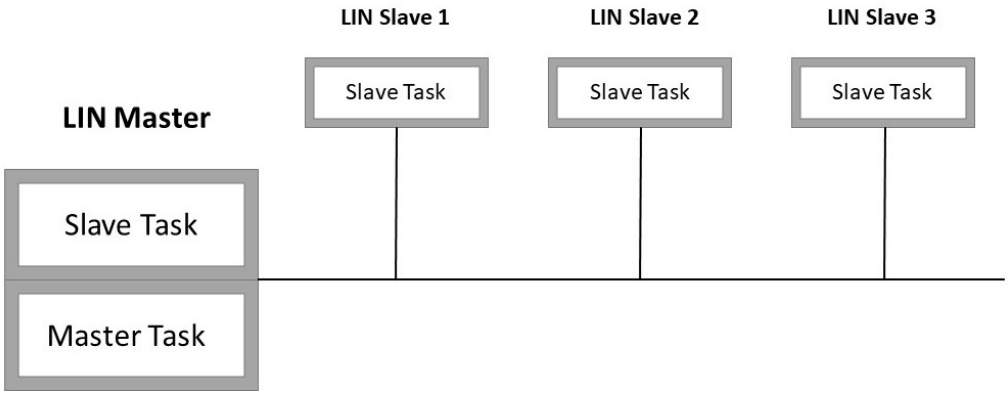

Figure 2.4. LIN bus.

Regarding to LIN frame, shown in Figure 2.5, the following fields are contained:

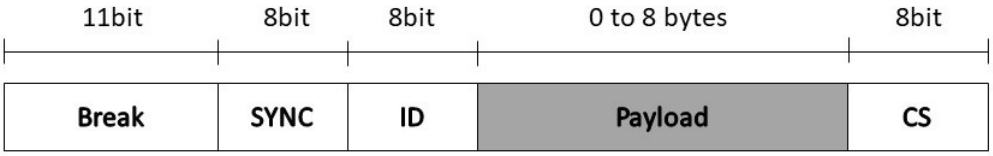

Figure 2.5. LIN frame.

- **Break**: this field informs all nodes that a message is transmitting. It is always transmitted by the master, as token;
- **SYNC**: field for synchronization. It is always transmitted by the master, as token;
- **ID**: message identification. The ID is univocally defined up to 64. It is always transmitted by the master, as token;
- **Payload**: data field. It can be transmitted by the slave or by the master;
- **Checksum**: 8 bits for error detections. It can be transmitted by the slave or by the master.

2.3 MOST

Around the 2000s, with the advent of first navigation systems, complex audio systems, Bluetooth, advanced displays, the data traffic is increased. Companies like BMW, Harman/Becker and SMSC formed the MOST Cooperation, developing the Media Oriented Serial Transport (MOST) technology. [67]
MOST has been developed for optimizing the transport of audio-video data because CAN and LIN networks can not guarantee high bandwidth for that purpose. In particular MOST150, the last MOST technology, can reach 150 Mb/s, supporting Ethernet Packet Channel with Internet Protocol (IP) functionalities.
MOST relies on all layers of OSI model. As physical layer, it usually runs on plastic optical fiber (POF) cable to avoid electromagnetic compatibility problems and to guarantee electrical isolation. However, MOST150 can support also coaxial cable. Both physical layers have advantages in terms of bandwidth, but they are too expensive. In fact, this reason and many other, shown later, led to a rapid decrease in the use of MOST technology.
Since this technology is synchronous based on Time Division Multiple Access (TDMA), all nodes have the same clock to avoid collisions.
MOST uses ring topology, as shown in Figure 2.6, with one ECU which works as timing master, allowing the synchronization, through synchronization messages, with all the other ECUs. Master ECU starts unidirectional communication with the next node, the latter will transmit the frame to next node, until it reaches the recipient. [4]

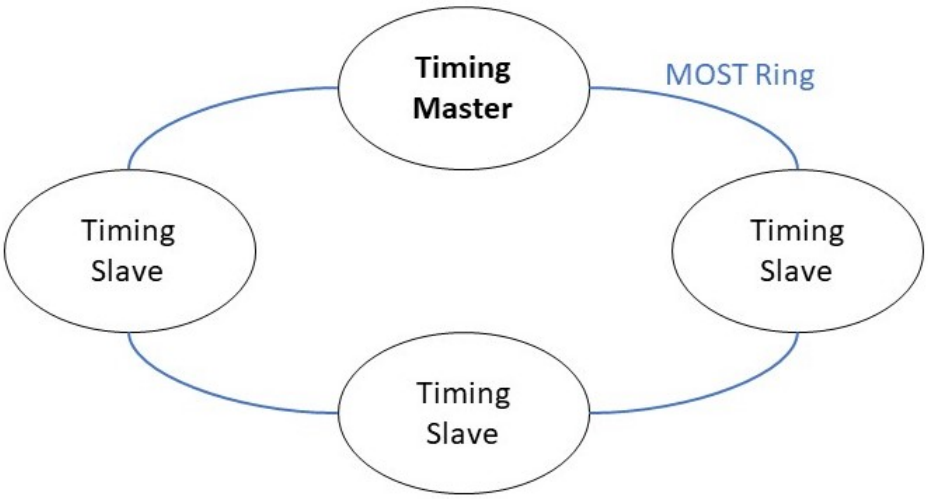

Figure 2.6. MOST ring topology.

In Figure 2.7 is shown a MOST150 frame with four main fields:

Figure 2.7. MOST150 frame.

- **Control Channel**: used for controlling network operations;
- **SYNC Channel**: used to exchange audio-video in a synchronous manner. (Analog and digital audio-video);
- **ASYNC Channel**: used to transmit packages asynchronously. (Internet traffic and information from the navigation system);
- **MOST Ethernet Packets**: introduced in MOST150 for IP data.

2.4 FlexRay

In the 2000s a consortium of major companies (BMW, Bosch, Daimler and many others) developed FlexRay protocol. The goal was to develop a time-triggered communication standard for safety-critical and time-critical automotive applications like all X-by-Wire systems (e.g. brake-by-wire, steer-by-wire and so on), powertrain and chassis controls.
As shown later there are many advantages and disadvantages in FlexRay, but many companies have stopped or are stopping using it, shifting their focus to Automotive Ethernet.
The technology is based on cycles, which are a combination of two types of windows: time-triggered (static) and event triggered (dynamic). The first one uses a Time Domain Multiple Access (TDMA) protocol, where there are many identical slots divided in time. If a node does not have to transmit, a null frame is sent, so that something is always received. [35]
The second one uses Flexible Time Division Multiple Access (FTDMA), where time is divided in sub-slots and each station can initiate communication within its sub-slot.
Regarding to the topology this network is very flexible, it can be used: linear bus, active and passive stars and point to point, as shown in Figure 2.8 and Figure 2.9. Topology can be also redundant using dual channels. [40] [67]
A FlexRay frame format, shown in Figure 2.10, consists in three parts: header, payload and footer section. In the header part there are:

- **Status Bit**: it represents the start of the frame which can be null frame, sync frame, startup frame;

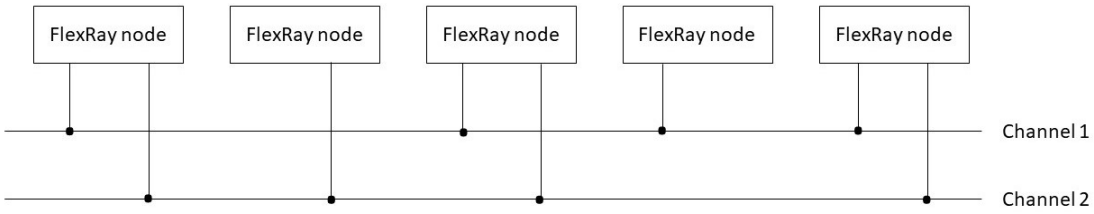

Figure 2.8. FlexRay linear bus topology.

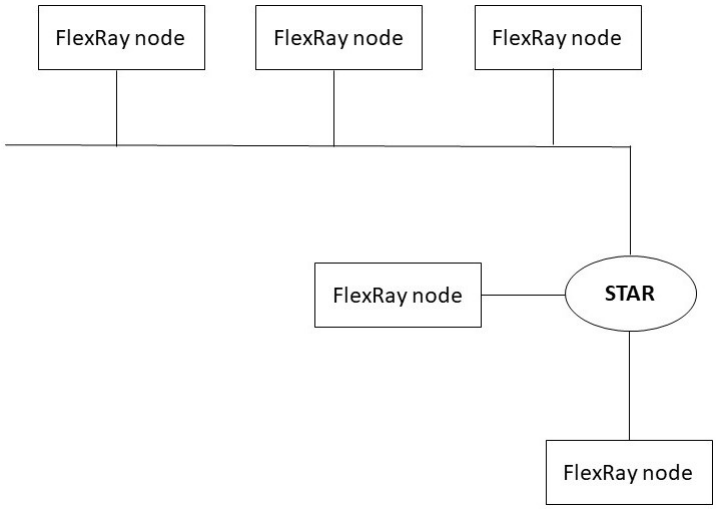

Figure 2.9. FlexRay star topology.

- **Frame ID**: unique identifier of the frame;
- **Length**: length of the data section;
- **Header CRC**: error checking of the header section;
- **Cycle**: number of the cycle.

Instead, the footer section is composed by the CRC field for checking errors in the payload.

2.5 Comparison

A small summary to compare all the network technologies previously analyzed, with respect to Automotive Ethernet, is shown in Table 2.1.

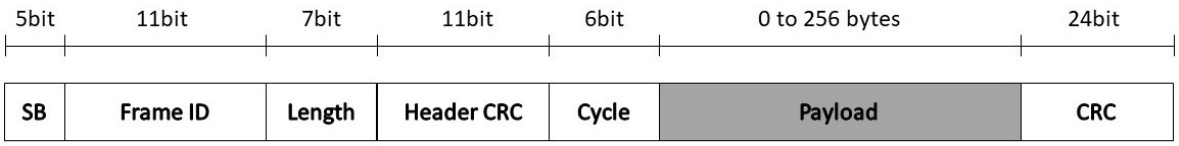

Figure 2.10. FlexRay frame.

Technology	Advantages	Disadvantages
CAN	- Low cost; - Robustness, safe, reliable; - Error capabilities: error counters, CRC fields; - Flexibility: CAN bus is multi-master, there is no need to add special purpose node (like switches), so a new node can be easily added or removed [4]; - Simulators and tools simple to use.	- Reduced data rates: between 125Kbit/s and 5-10Mb/s for CAN-FD and CAN-XL; - Low efficiency: very low payload 8 byte for classic CAN, 64 bytes for CAN-FD, consequently high overhead.
LIN	- The lowest cost of all network technologies; - Easy to implement.	- Reduced data rates: up to 20 Kbit/s; - The messages are scheduled in time. Consequently, the master can not start immediately a communication; - Low error detection capabilities.
MOST	- Very high bandwidth (150 Mbit/s) respect to CAN, LIN; - Immunity to electromagnetic compatibility problems.	- Too expensive due to optic fiber and coaxial cable; - Low flexibility; - Silicones for MOST are only developed by one company.
FlexRay	- Dual redundant channels; - Flexible in topologies.	- Bandwidth of only 20 Mbit/s; - No protocol support for infotainment; - Complicated to implement.
Automotive Ethernet	- Very high data rate: 10Mbps to 10Gbps; - Robustness: differential signal, smart modulation and filtering; - Flexibility in topologies; - Many protocols, standards and applications; - Reductions in vehicle weight and cost.	- Cost of new technology higher than a more mature one (it will settle down), plus the cost of switches; - Adding flexibility means adding cost for switches; - Test tools are not so obvious. They will be analysed in a future chapter.

Table 2.1. Network technologies comparison.

Chapter 3

Automotive Ethernet

This chapter introduces, firstly, Ethernet used in local area networks, then the relevant subject to this thesis: Automotive Ethernet with the differences and integrations each other. It will be explained the influence of Ethernet in the automotive field, all the OSI model layers, from physical to application layer, with the main dedicated Application Protocols.

3.1 Standard Ethernet

Ethernet network, the most used Local Area Network (LAN) in the word, was developed in the early 70's by Xerox PARC which needed a connection between computer, internet and printer. Xerox opened the technology to everyone, consequently, in 1983, the 802.3 Project, inside the Institute of Electrical and Electronics Engineers (IEEE), was created. Over the years IEEE 802.3 added, and continues to add, new Ethernet Standards. The IEEE 802 project structure defines three layers, as shown in Figure 3.1: [48]

- **Physical Layer (PL)**:
  - *10 Mbit/s Ethernet*: it uses the Carrier Sense Multiple Access with Collision Detection (CSMA/CD)[1] protocol;
  - *Fast Ethernet*: evolution of the previous generation. It uses a full-duplex transfer and a star center (switch) configuration. In fact, each received frame is retransmitted only to the recipient line and not to all;

[1]CSMA/CD: protocol that allows to detect any collisions in the network. Carrier sense listens to the channel before transmitting, if it detects no signals it transmits. Collision detection checks for collisions during transmission. If a collision is detected the transmission is interrupted and is delayed after a random time established by a backoff algorithm.

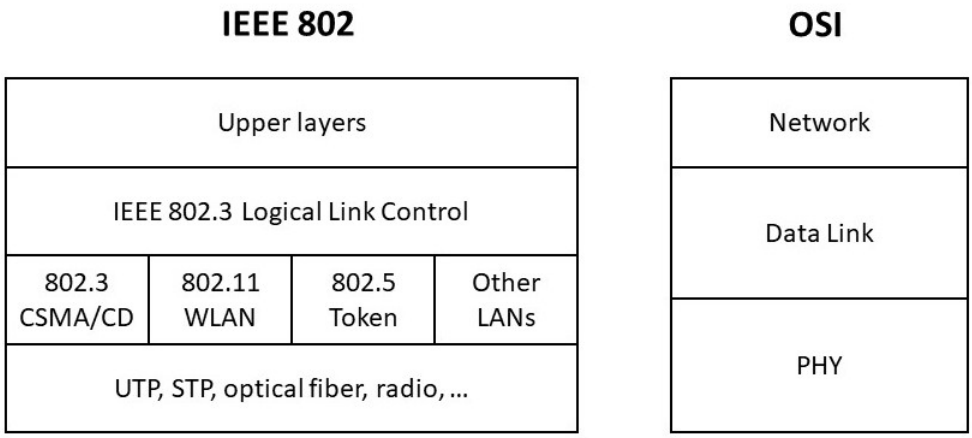

Figure 3.1. IEEE 802 project levels structure.

  - *Gigabit Ethernet*: evolution of Fast Ethernet with the possibility of using half-duplex transmissions with hub or full duplex with switch;
  - *10 Gigabit Ethernet*: evolution exclusively full duplex.

  There are different types of networks, XBaseY, depending on the bandwidth X, length of a network segment and type of transmission medium Y. For example 10-Base-T is a 10 Mbit/s network on unshielded twisted pair (UTP).

- **Medium Access Control (MAC)**: it is responsible for how the nodes access the network, to control packet losses and how to handle a collision;
- **Logical Link Control (LLC)**: it hides the type of MAC protocol used (lower layer) and works as an interface to the upper layer (network);

According to IEEE 802.3 standard, the Ethernet frame, shown in Figure 3.2, is composed of the following fields:

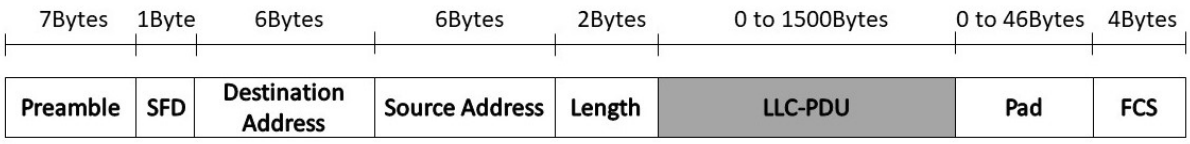

Figure 3.2. IEEE 802.3 Ethernet frame.

- **Preamble**: used to synchronize the signal between origin and destination;

- **Start of Frame Delimiter**: it is a 10101011 sequence which indicates the start of frame;
- **Destination and Source Address**: indicate address of the recipient and sender station respectively;
- **Length**: length of the next field;
- **LLC-PDU**: payload of the transmitted data;
- **Pad**: it guarantees a minimum length;
- **Frame Check Sequence**: it contains the Cyclic Redundancy Check (CRC) to check errors.

3.2 From Standard Ethernet to Automotive Ethernet

As anticipated in the introduction, IEEE 802.3 Ethernet was born to guarantee very high speed, but as it was, it could not guarantee strictly requirements in terms of latency, jitter, congestion, guaranteed bandwidth and message arrivals, consequently it could not be used for real-time or multimedia applications. Moreover traditional Ethernet is too noisy and it has too interference for automotive applications.
For these reasons, the automotive industry has taken some parts of IEEE 802.3 Ethernet and changed others. For weight and cost reasons the 4-pair single direction cable is substituted by a single pair bi-directional UTP cable for both transmit and receive. Mainly the changes are in the physical layer using, as it will be seen, the Broadcom BroadR-Reach technology, promoted by OPEN Alliance, which permits full-duplex communication, ideally doubling the throughput, unlike CAN, LIN, Most, FlexRay.[4] Main differences are shown in Table 3.1.
Besides data link and physical layers also other layers have to be taken into account, from network layer to application layer.
The first vehicle with Automotive Ethernet was released in 2008 by the BMW Group for diagnostic purposes. Later in 2013 the new BroadR-Reach 100Base-T1 physical layer was used in BMW's X5.
For this purpose, in time, three Automotive Ethernet use cases (generations) can be considered: [13]

- **First generation** ⇒ *Diagnostics Over IP (DoIP)*: used to synchronize the signal between origin and destination;
- **Second generation** ⇒ *ADAS and Infotainment*: for camera systems and infotainment applications;

	Ethernet 100Base-TX *(IEEE 802.3)*	**Automotive Ethernet 100Base-T1** *(IEEE 802.3bw)*	**Automotive Ethernet 1000Base-T1** *(IEEE 802.3bp)*
Data Rate	100 Mbps	100 Mbps	1000 Mbps
Signal	MLT3	PAM3 at 66.667 Mb/s	PAM3 at 750 Mb/s
Modulation	4B/5B	4B/3B, 3B/2T	80B/81B
Length	100m	15m	15m
Connector	RJ45	Depends on the manufacturer	Depends on the manufacturer
Cable	Two twisted pairs single direction	One twisted pair bi-directional	One twisted pair bi-directional

Table 3.1. Main differences between Ethernet and Automotive Ethernet. [32]

- **Third generation** ⇒ *Ethernet Network Backbone*: communication between ECUs in a hierarchical way as a scalable solution with different data rate.

Consequently, new protocols for different use cases (generations) have been developed, as shown in Figure 3.3.

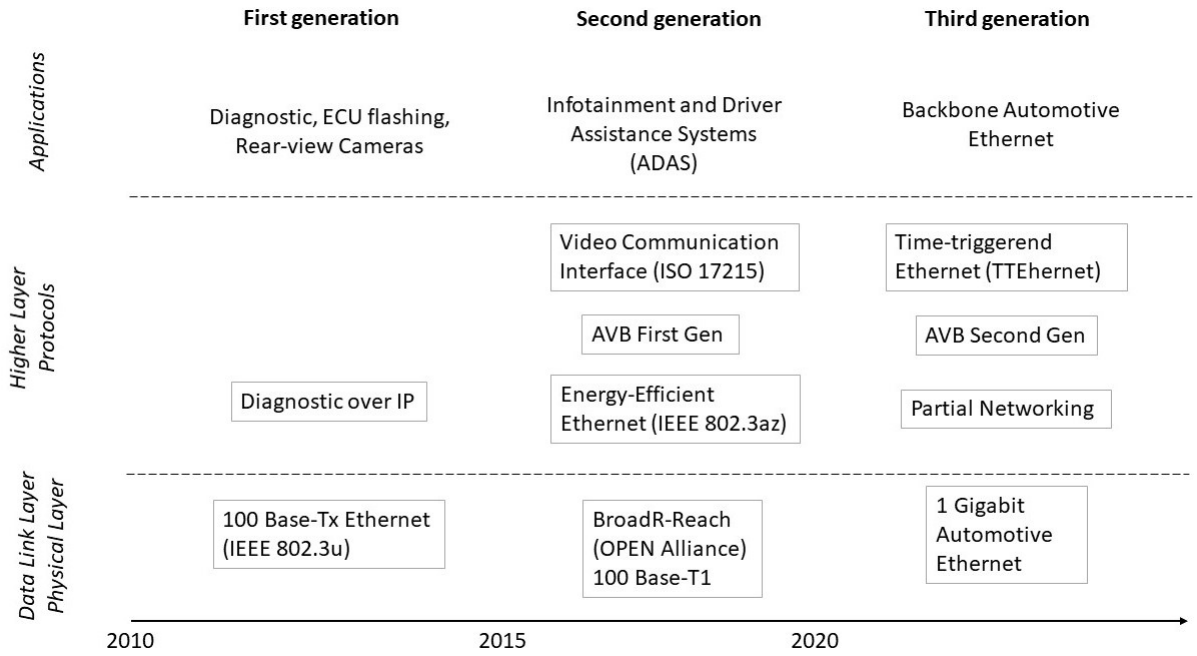

Figure 3.3. Protocols for different use cases (generations) of Automotive Ethernet. Source: [27]

In the following sections, according to OSI model, will be covered in detail:

– Physical layer technologies for 10 Mbps, 100 Mbps and 1 Gbps;

– MAC frame and VLAN;

– Internet Protocol (IP);

– TCP/UDP;

– Dedicated Application Protocols (SOME/IP, AVB/TSN, TTEthernet, DoIP).

3.3 Physical Layers

To overcome the bandwidth issue, in these years, many Ethernet standards have been developed. In this section they will be analysed with emphasis on different data rates: 10 Mbps, 100 Mbps and 1 Gbps. In Figure 3.4 is shown a time horizon of the various standards.

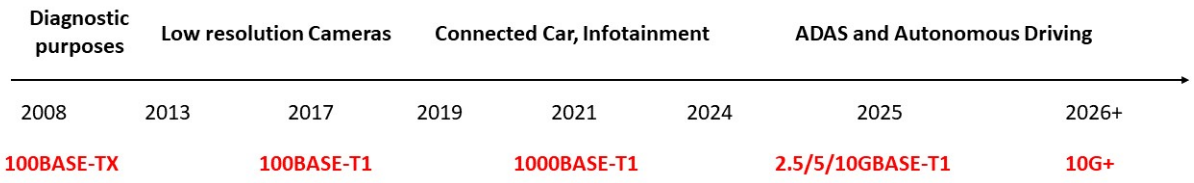

Figure 3.4. Time horizon of Automotive Ethernet Standards.

3.3.1 100 Mbps and 1000 Mbps networking technologies

100BASE-T1 The most used Fast Ethernet standard for local area networks, 100BASE-TX, is based over two pairs of unshielded twisted pairs (UTPs). One pair is used for transmission and the other one for reception, establishing a 100 Mbps full-duplex communication. Instead, 100BASE-T1, born as BroadR-Reach and standardized by IEEE as 802.3bw, provides well established limits on EMC, EMI, temperature-grade and uses a single UTP cable, as shown in Figure 3.5, for bidirectional signal (meaning low cost and low cabling weight).
Differences between the two communication methods are shown in Figure 3.6.
Respect to 100BASE-TX (standard Ethernet cable), BroadR-Reach cable can achieve 80% cost reduction and 30% weight reduction according to Open Alliance and Broadcom publications. For these reasons 100BASE-T1 has taken over, particularly in the automotive field. As it will be seen 100BASE-T1 can also support communication of audio/video using AVB/TSN standard, it can support different data types with many priorities.
Comparing the requirements of 100BASE-T1 on Electromagnetic Interference (EMI)

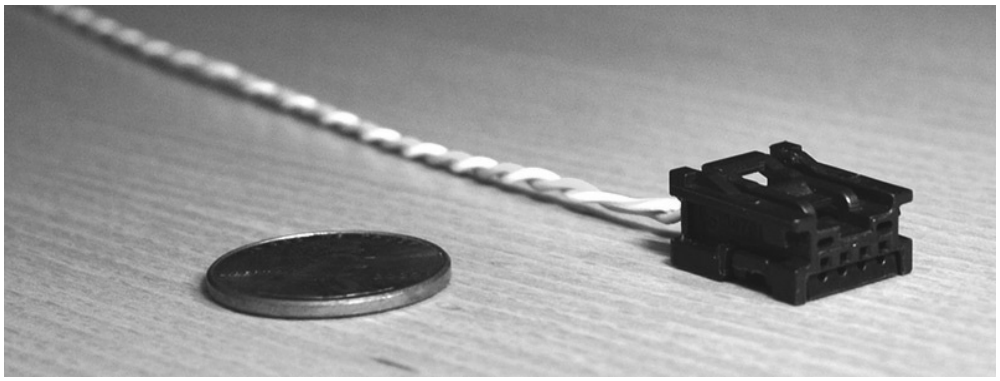

Figure 3.5. BroadR-Reach cable and connector. Source: [4].

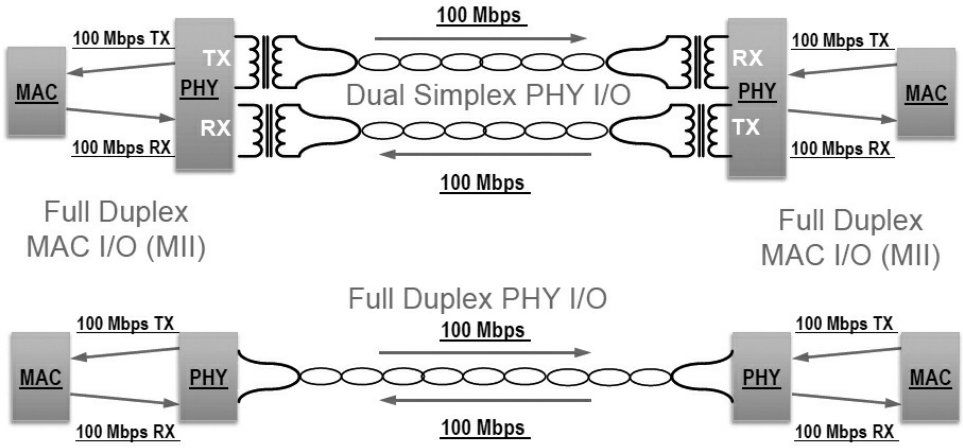

Figure 3.6. Difference between 100BASE-TX and BroadR-Reach 100BASE-T1 communications. Source: [4].

and Electromagnetic compatibility (EMC) with respect to CAN interface, which is currently the most widespread technology, as shown in Figure 3.7, both are below the maximum limits imposed by the Comité International Spécial des Perturbations Radioélectriques (CISPR) 25 Class 5 Annex G [5] and TC8 Open Alliance [44].
Another difference between 100BASE-TX and 100BASE-T1 regards to signaling and modulation. The first one uses Multi-Level Transmit MLT3 (tension levels: -1, 0, +1, 0) with 4bit/5bit as modulation on 125 Msymbols/sec.
Instead the second one uses Pulse Amplitude Modulation PAM3 (three tension levels: +1, 0, -1) with 4bit/3bit, 3bit/2ternary symbols, as modulation on 66.66Msymbols/sec. This to achieve $66.66 Msymbol/s * 3B/2T = 100 Mbps$. A coding example on 4B/3B, 3B/2T, PAM3 is shown in Figure 3.8.

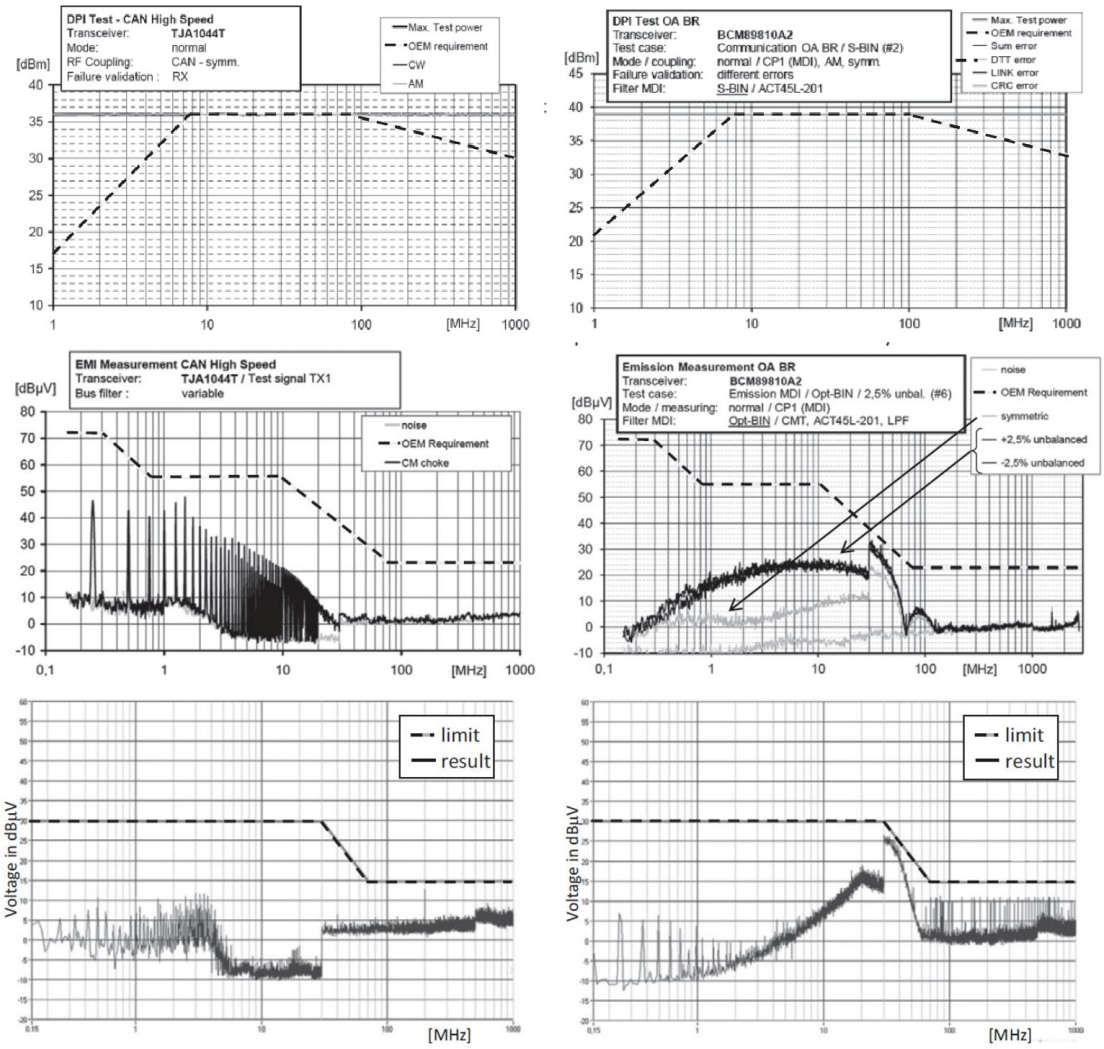

Figure 3.7. Emission tests. Source: [35].

As said, since 100BASE-T1 uses a single pair cable, hybrid circuit and echo canceler, to distinguish between transmitted/received signals are needed, as shown in Figure 3.9.
The hybrid circuit permits the two-way communication on the same wire pair with master and slave principle with a handshake process for start-up. This process uses signals:

- **SEND_Z**: transmission of all zeros;
- **SEND_I**: transmission of PAM3 (-1V, 0V, 1V) idle signals;
- **SEND_N**: transmission of PAM3 data signals or idle signals.

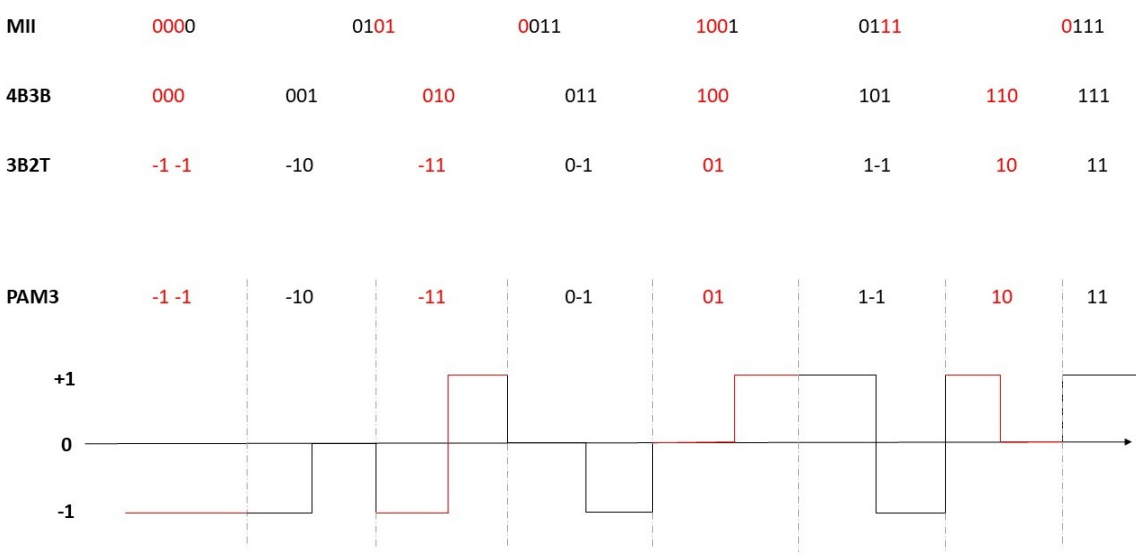

Figure 3.8. Coding example on 4B/3B, 3B/2T, PAM3.

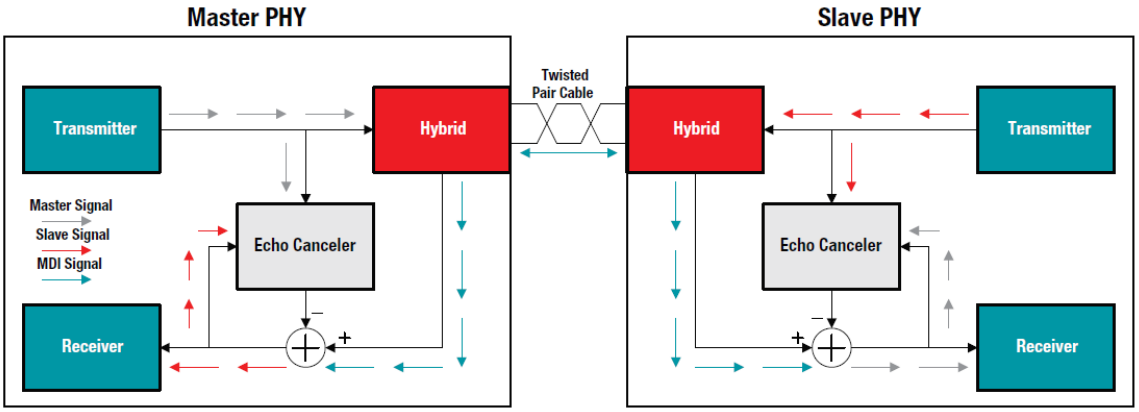

Figure 3.9. Echo cancellation and hybrid block diagram. Source: [49].

Initially, the Master goes from SEND_Z to SEND_I state, transmitting PAM3 idle signals, at the same time the Slave remains on SEND_N state. Then, the Slave goes to SEND_I state with Master on SEND_I. Finally, if Master and Slave validate the start-up process, both go to SEND_N state, however the process restart. An example of process is shown in Figure 3.10. [60]
A typical 100BASE-T1 implementation is shown in Figure 3.11 where, instead of transformers (case of 100BASE-TX), two capacitors to reduce the dimensions are placed.

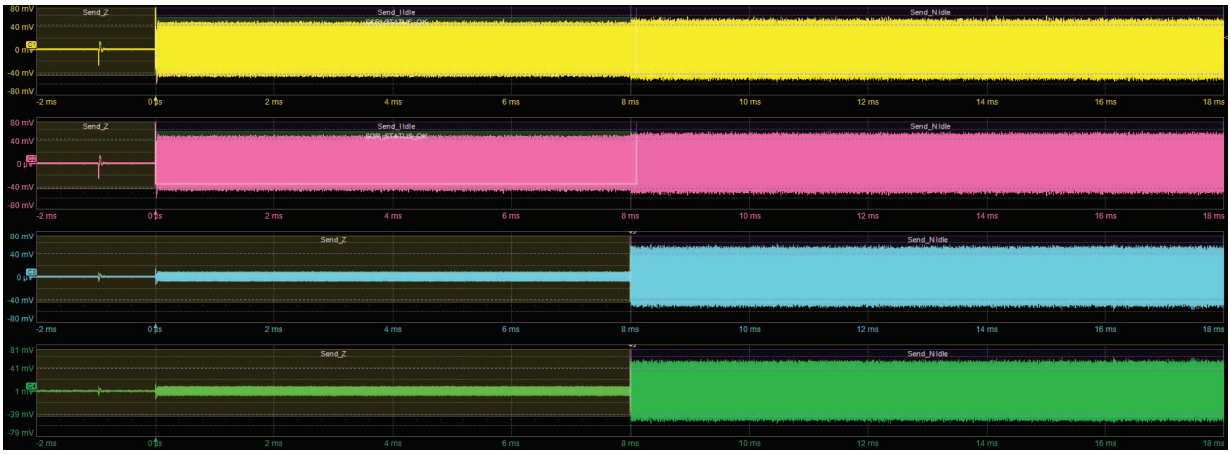

Figure 3.10. Link startup process: Master (yellow, pink) and Slave (blue, green). Switch between SEND_Z, SEND_I, SEND_N. Source: [60].

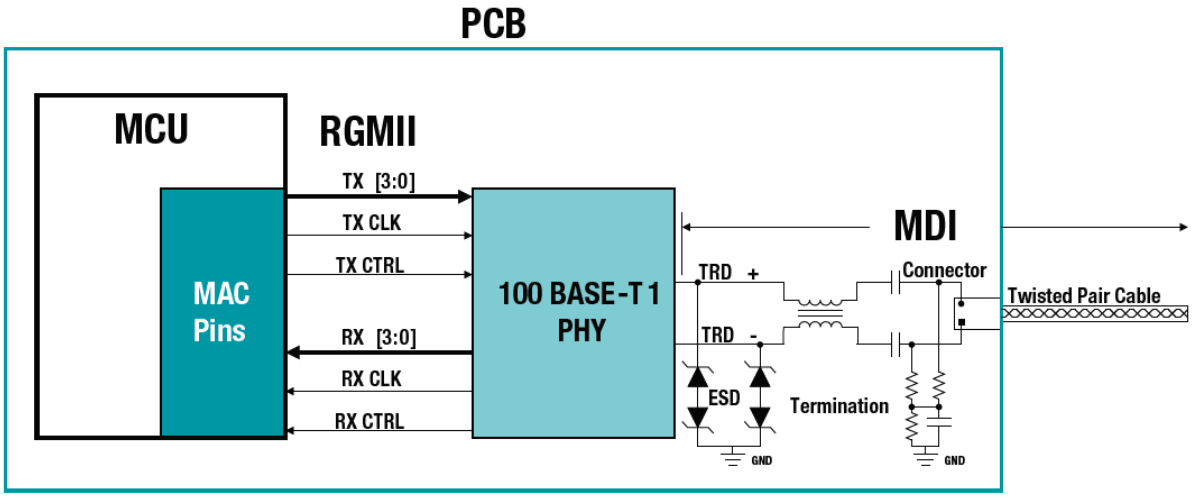

Figure 3.11. 100BASE-T1 implementation. Source: [49].

Regarding to the 100BASE-T1 frame, is practically the same as IEEE 802.3 standard Ethernet, shown in Section 3.1, with Start of Stream Delimiter (SSD) at the beginning and END of Stream Delimiter (ESD) at the end of frame. [60]

1000BASE-T1 The bandwidth increasing, the birth of autonomous driving, the use of many sensors and the need for redundancy, during the years, led to switch to a 1Gbps technology such as 1000BASE-T1 standardized by IEEE 802.3bp.
1000BASE-T1 uses Pulse Amplitude Modulation PAM3 (three tension levels: +1, 0, -1) with 3bit/2ternary symbols as modulation on 750Msymbols/sec, this to achieve

$750Msymbol/s * 3B/2T = 1000Mbps$.

3.3.2 10 Mbps networking technology

10BASE-T1S With the advent of new sensors applications and ECU developments, a 10Mbps Ethernet PHY was born that can be used for:

- ECUs that require faster communication than CAN-FD, but lower then 100Mbps of 100BASE-T1 for cost and energy reasons;
- Simple, redundant sensors and actuators network (only 1 connector at ECU per bus line like shown in Figure 3.12);
- As a replacement for legacy networks such as FlexRay.

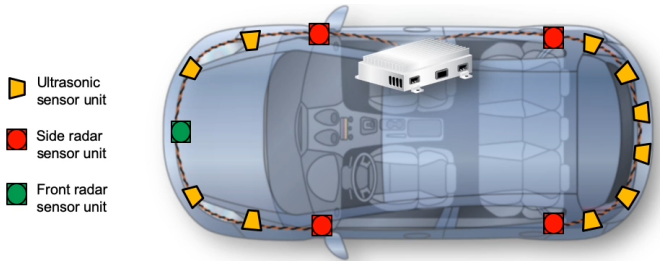

Figure 3.12. Sensor network example on 10BASE-T1S.

In particular IEEE 802.3cg defines PHY 10BASE-T1S, a low-complexity pair, where "S" means short length of 15-25m (there is another PHY 10BASE-T1L for long distance, 1km, but not useful for automotive applications).
10BASE-T1S is on single twisted pair and uses 4B5B encoding, its peculiarity is that it supports three different operating modes: [72]

- Point-to-point 15m with optional full duplex;
- Point-to-point 15m with mandatory half duplex;
- 25m optional half duplex multidrop, it supports bus architectures (like CAN), using Physical Layer Collision Avoidance PLCA (a CSMA/CD extension) to improve performance in stress condition.

Full duplex and multidrop options are in mutually exclusive.
As said the principal reason for using 10BASE-T1S is the cost, due to the multidrop topology and a simplified PHY design. In case of a multidrop topology only one PHY per end node ECU is used, instead of two for traditional Ethernet, less connectors and switches. This saves $(N-2)/(2N-2)$ PHYs with N number of

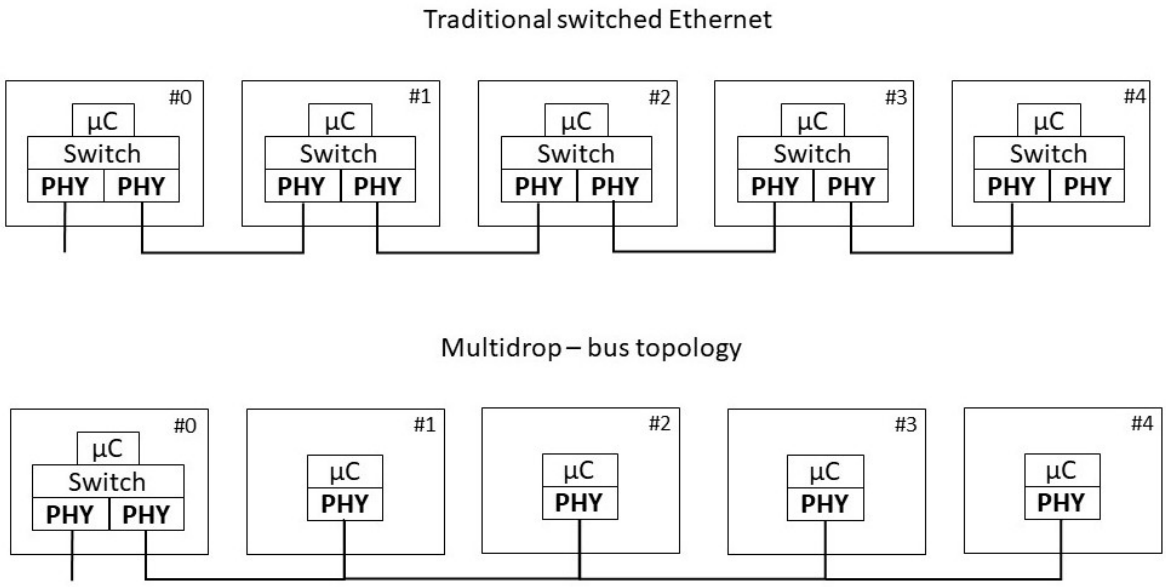

Figure 3.13. Difference between traditional switched Ethernet and 10BASE-T1S as multidrop topology.

nodes. [35] The difference is shown in Figure 3.13.
In Figure 3.14 cost and data rate are compared with the target that will be reached.

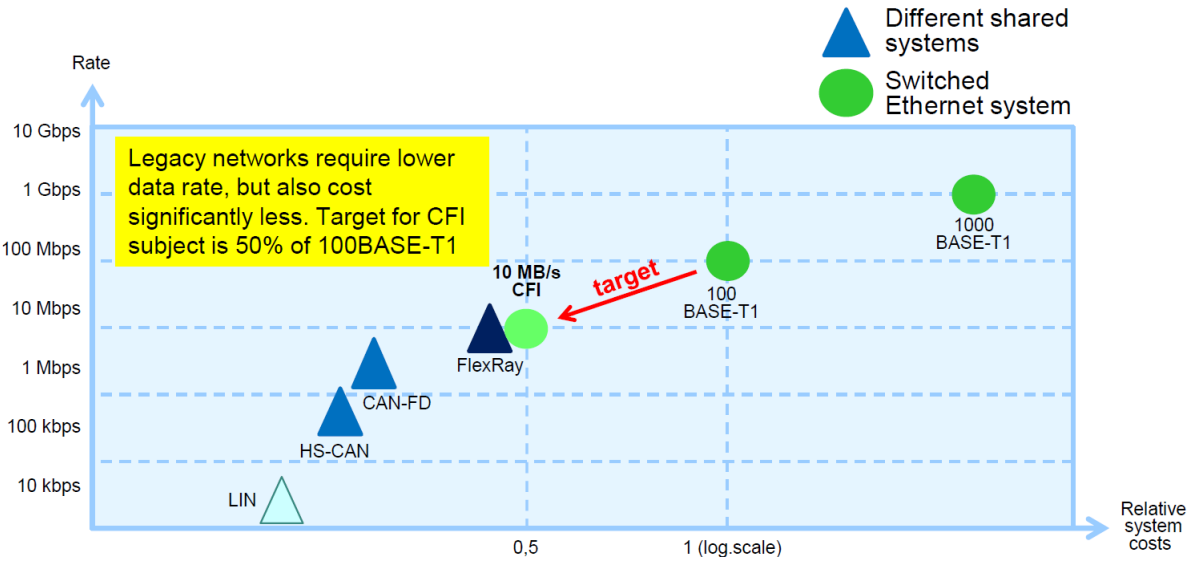

Figure 3.14. Comparison between cost and data rate of different technologies. Source: [14]

PLCA can offer easily and bounded channel-access time by means of a Round-Robin schedule that it does not allow collisions and permits throughput of about 10Mbps in stress condition. In Figure 3.15 is shown the PLCA mechanism, where the units transmit their own packets in sequence of the node IDs. The first unit, head node, that has ID equal to zero, starts every cycle with a signal (Beacon) that inform all the other nodes to restart timers and counters. For transmitting every unit uses its Transmit Opportunity (TO), if it does not use it, the unit with the increasing ID can start when the TO time expires. [35]

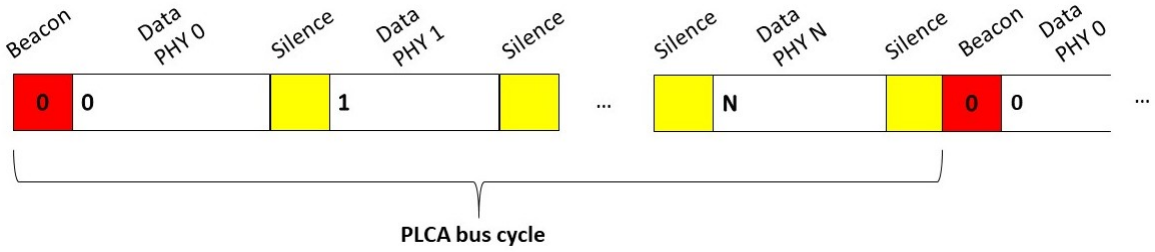

Figure 3.15. PLCA mechanism.

PLCA being an extension of CMSA/CD, a comparison of bandwidth and access latency between CSMA/CD + PLCA and CSMA/CD only is shown in Figure 3.16.

10BASE-T1 has not yet been implemented on vehicles, but extensive tests have been done by Open Alliance members.

3.4 IEEE Ethernet MAC and VLAN

While the previous section dealt with physical layer, here we will move one level on the stack architecture, dealing with the Media Access Control (MAC) sublayer. Since two or more devices can access simultaneously to the bus, collisions can occur. All Ethernet controllers have a collision detection function to stop a transmission. To prevent another collision, a node can transmit only after the expiration of a random time thanks to a backoff algorithm. The compete method to access the bus is called Carrier Sense Multiple Access Collision Detection (CMSA/CD), talked about in the previous sections. However, in our case, since IEEE 100BASE-TX, IEEE 100BASE-T1 and 1000BASE-T1 use a full duplex communication, collisions do not occur in these physical media.
In order to send data and to communicate between nodes, they must have an identifying address that is unique. In IEEE 802 scheme, the MAC address is defined in six hexadecimal bytes with the first three assigned to each hardware manufacturer and the other three used by each manufacturer to assign the address to their products.

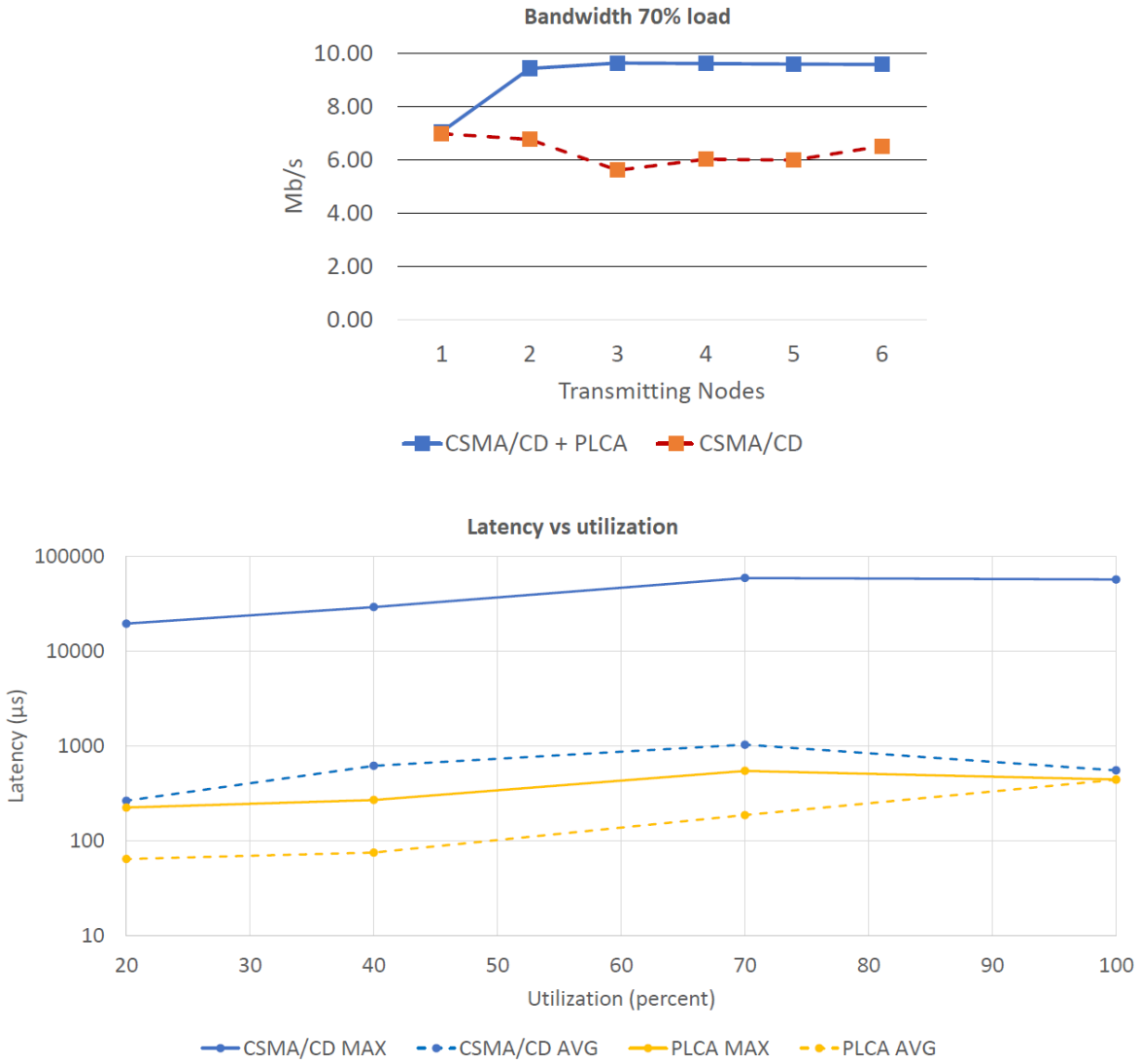

Figure 3.16. Comparison of bandwidth and access latency between CSMA/CD + PLCA and CSMA/CD only. Source: [72]

To extend the classic addressing Virtual Local Area Network (VLAN) can be used, especially in automotive applications. VLAN addresses permit to create virtual networks on top of physical network and to delimit communications, defining domains for different applications and use cases. An ECU can be part of more than one application domains and VLANs. The Virtual LAN is also a basis for security reason due to its firewall function, because, for instance, a certain browser application cannot access to car internal data, even if the data goes on the same wires.

In Figure 3.17 is shown a VLANs example where, for instance, if diagnostic interfaces are considered, the diagnostic packets receive the diagnostic VLAN tag. Inside a car, this traffic is recognized as diagnostic traffic and not as car internal traffic. [35]

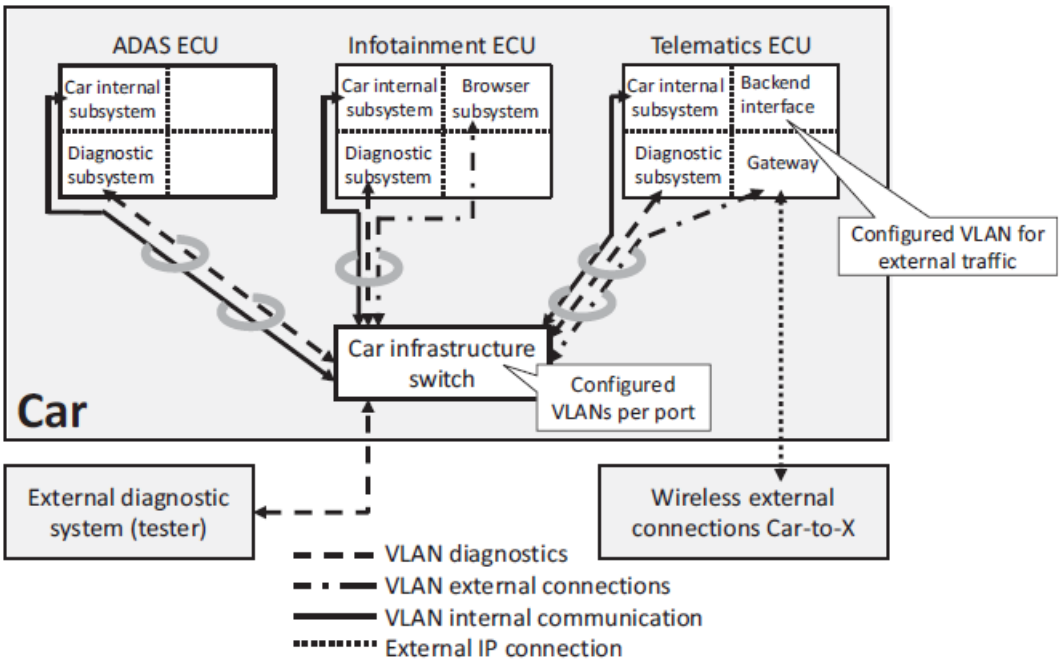

Figure 3.17. VLANs example. Source: [35]

It is also important to point out that VLAN permits to have different priorities for each message routed to the Ethernet switches for better real-time communications. A transmitted message always contains an origin address and a destination address. As specified in previous sections, in automotive field, Ethernet frames II, shown in Figure 3.18, are used. See Section 3.1 for further insights.

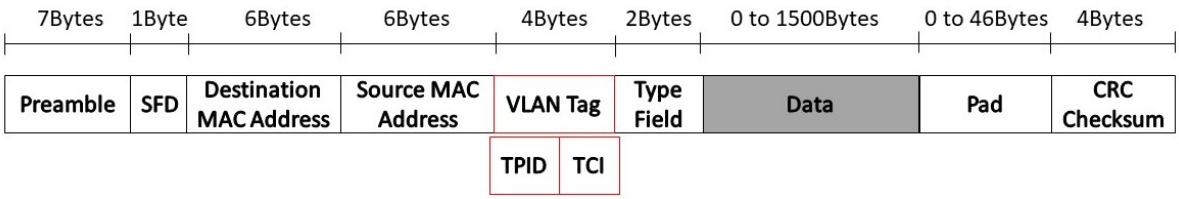

Figure 3.18. Ethernet frame with VLAN Tag.

In automotive field applications the frame contains also VLAN extension in the VLAN Tag that contains:

- **TPID**: Tag Protocol identifier, 2 byte that highlights the frame format;
- **TCI**: Tag Control information, 2byte divided into:
  - *PCP*: Priority Code Point, to indicate a priority level of the frame;
  - *DEI*: Drop Eligible Indicator, to have a possibility of ignoring the frame in case of congestion;
  - *VEI*: VLAN ID, to indicate the VLAN number between 0 and 4096.

3.5 Internet Protocol IP

There are no particular differences regarding Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Internet Protocol (IP) in automotive industry, in fact the ability to reuse all protocols is a great advantage.
The IP protocol provides a network message transfer service without previously establishing a connection and without providing guarantees on the actual transfer of IP datagrams. The essential functionality of IP is addressing, identifying and locating hosts and packages from source to destination address.
An IP datagram is composed by a header and a payload. In case of IPv4 (32bit), the header, shown in Figure 3.19, contains the following fields: [48]

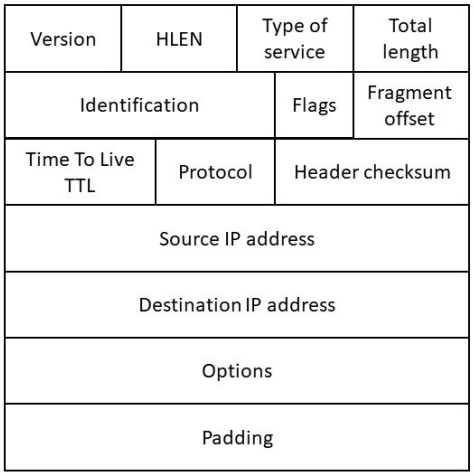

Figure 3.19. Header part of IP datagram.

- **Version**: it specifies the protocol IP version (4 in this case);
- **HLEN**: header length in multiples of 32 bits;
- **Type of service**: it specifies how and precedence the receiving host should treat the datagram;

- **Total length**: total length, in byte, of the IP datagram;
- **Identification**: it identifies univocally the datagram (if fragmented);
- **Flag**: 3 bits used for protocol and datagram fragmentation control;
- **Fragment offset**: offset, in 8 bytes, of a particular fragment relative to the beginning of the original IP packet;
- **TTL**: Time To Live of the packet, necessary to avoid indefinite persistence on the network in case it is not possible to deliver it to the recipient;
- **Protocol**: higher level protocol that originated the datagram;
- **Header checksum**: checking for header errors;
- **Source IP address**;
- **Destination IP address**;
- **Options**: for more specific uses of the protocol;
- **Padding**: filling field to ensure multiple 32bit header length.

IPv6 uses 128bit instead of 32bit for addresses, but currently solutions with IPv6 are not mature enough yet.

Even if the number of active nodes can change, an in-vehicle network is a closed system because the maximum number of nodes is established a priori. Since a car have to be restarted many times during a day, there is a time issue which requires a quick start of all nodes, therefore static IP configurations are recommended. However, there are also some applications that require dynamic IP.
To assign IP addresses inside a vehicle four methods can be used: [35]

- **Static**: The IP addresses are assigned during the development phase of the ECU, consequently each ECU with the same function will have the same IP address regardless of the car on which it is implemented. Obviously, there must not be two ECUs with the same function and consequently IP;
- **Pseudo-dynamic**: The ECU produced has no IP address, it will receive it statically during assembly;
- **Dynamic**: Required when the vehicle communicates with the outside world, for instance, for diagnostic tester;

- **Multiple**: A single ECU uses many IP addresses, for instance, for diagnostic purposes where if the address was single, it had to be changed every time it was connected with external world. An example is shown in Figure 3.20 where the dynamic IP addresses y and z are assigned only when the External diagnostic system (tester) is connected with dynamic IP address x.

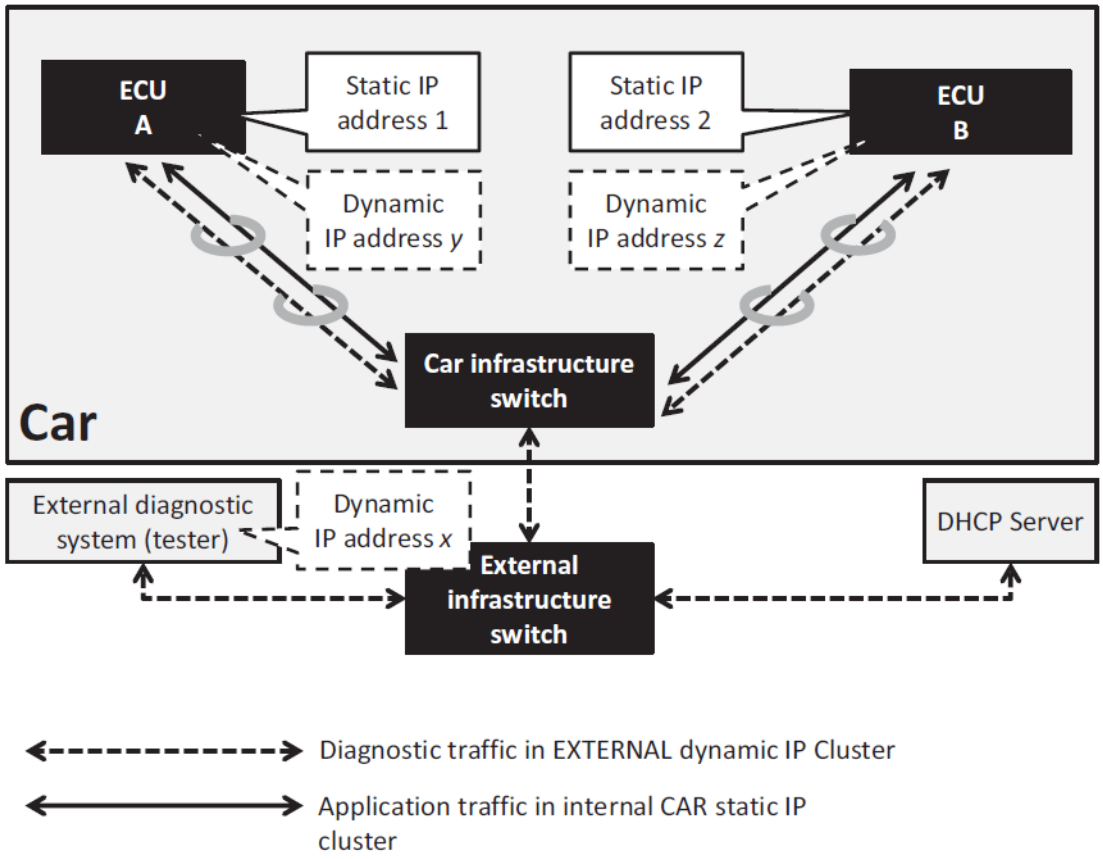

Figure 3.20. Example of multiple IP addressing. Source: [35]

The switches of OSI layer two, using MAC addresses have lower latency and can only be implemented on hardware, compared to a more complex, larger and expensive in software, router of OSI layer three. Consequently, currently dedicated router chips for the automotive field are not available, contrary to switches. [35]

3.6 TCP/UDP

At transport layer the main protocols used are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). The first one is connection-oriented

and the second one is connectionless. Both protocols carry segments, called Transport Protocol Data Units (TPDU). Since the lower IP layer carries units of various applications, ports, acting as an interface between the applications and the IP layer, are defined. Port number and IP address make up a socket.

UDP protocol The objective of User Datagram Protocol (UDP) is to multiplex more information flows on a single UDP flow, checking for errors only at the nodes. The transfer is connectionless, so out of sequence deliveries can happen and there is no connection establishment. It is also not possible to perform segment loss checks, as they are not numbered.
However, having connectionless transmission results in faster transmission than TCP protocol. It is also possible to send segments in Multicast or Broadcast mode. A header of an UDP segment is shown in Figure 3.21, where the following fields appear: [48]

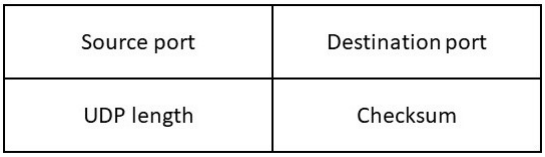

Figure 3.21. Header of UDP segment.

- **Source port**;
- **Destination port**;
- **Length**: in bytes of the entire UDP segment;
- **Checksum**: error detection for the UDP segment.

An example for data transmission between two ECUs, in UDP protocol, is shown in Figure 3.22 where it is possible to see that acknowledgments are not generated.

TCP protocol The objective of Transmission Control Protocol (TCP) is to improve the underlying connectionless IP protocol service by providing a reliable connection-oriented service, multiplexing multiple streams into a single TCP stream.
In contrast to UDP, in case of loss, duplication or out of sequence deliveries segments, the TCP protocol reconstructs the flow thanks to an appropriate numbering of the segments. In this case Broadcast or Multicast mode are not possible to use, because the connection must be between two ECUs.
A header of an TCP segment, which proves its reliability, is shown in Figure 3.23 where the following fields are used: [48]

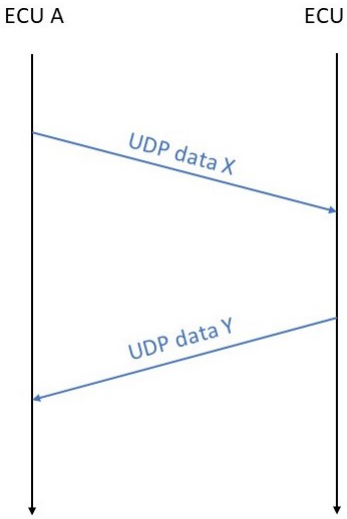

Figure 3.22. Example of UDP data transmission.

Source port			Destination port
Sequence number			
Acknoledgment number			
HLEN	Reserved	Code bits	Window
Checksum			Urgent pointer
Options			
Padding			

Figure 3.23. Header of TCP segment.

- **Source port**;
- **Destination port**;
- **Sequence number**: order number of the first data byte of the segment;
- **Acknowledgement number**: order number of the next byte that the communication expects to receive with the next message;

- **HLEN**: header length;
- **Code bits**: 6 bits of flags:
  - *URG*: it indicates the presence of urgent data;
  - *ACK*: it indicates whether the matching number is valid or not;
  - *PSH*: informs the receiver to deliver the data to the application; immediately, regardless of the buffer;
  - *RST*: connection reset, connection refusal or message;
  - *SYN*: to establish a TCP connection;
  - *FIN*: to release a TCP connection.
- **Window**: number of bytes the source can accept (flow control);
- **Checksum**: error checking;
- **Urgent Pointer**: location (offset) of urgent data;
- **Options**: particular functions.

The connection between two nodes, as in Figure 3.24, is established through a three-way handshake procedure where, firstly, an ECU A sends request for connection establishment for ECU B, specifying the Sequence Number. The ECU B sends the acceptance of the connection also specifying the sequence number. Finally, ECU A sends the final acknowledgement of acceptance.

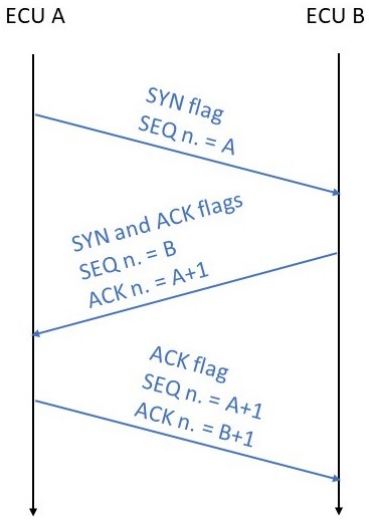

Figure 3.24. Opening TCP connection between two ECUs.

Regarding to data transmission of TCP segments, an example is shown in Figure 3.25. The ECU A sends the data X to ECU B, this answers with an acknowledgment and data Y. Finally ECU A acknowledges data Y.

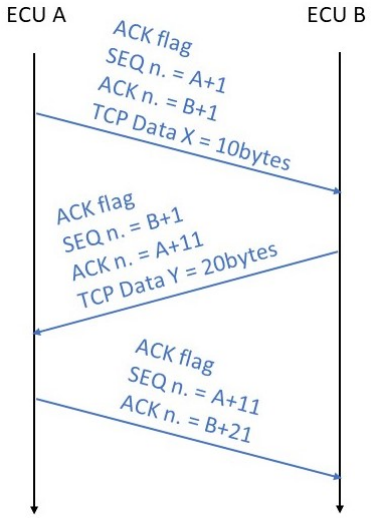

Figure 3.25. TCP data transmission.

At the end, the connection is closed, as in figure 3.26, where the ECU A sends the request with flag FIN, the ECU B acknowledges. Then also ECU B sends the request with flag FIN and also ECU A acknowledges.

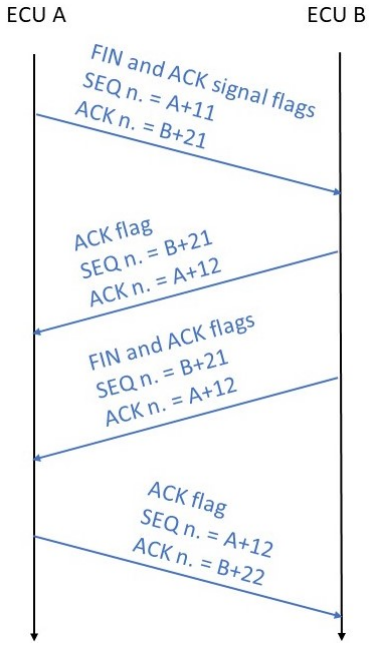

Figure 3.26. Closing TCP connection between two ECUs.

3.7 Application Protocols

In this subsection will be described the most used application-related protocol in automotive field, such as: SOME/IP, AVB/TSN, TTEthernet and DoIP.

3.7.1 SOME/IP

Scalable service-Oriented MiddlewarE over IP (SOME/IP) was designed by BMW group in 2011 and it was implemented in series production cars at BMW in 2014. From version 4.1 of AUTOSAR it has become an integral part of it providing scalability, flexibility and adaptability. However, in addition to AUTOSAR, it shall be implemented on different operating system like OSEK and even embedded devices without operating system.
SOME/IP works as middleware, like shown in Figure 3.27, in the sense that makes the network transparent to the software exchanging data, doing as an intermediary between different applications. The message communications and function calls between software are implemented just once to better test the modules.
SOME/IP allows applications to communicate providing service-oriented communication over a network, in this way the sender sends his message only when at least one recipient has requested it so that the network does not have to process useless data. On the contrary, in a signal-oriented transmission the data is sent by

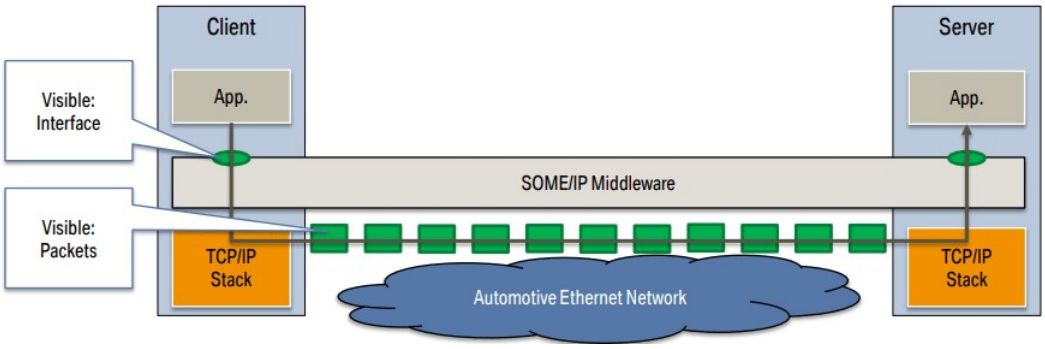

Figure 3.27. SOME/IP middleware. Source [9].

the sender when values are updated regardless of the need of the receiver.
SOME/IP is defined on client-server architecture, there are several communication methods between client and server: [35][9]

- **Request and Response**: the client sends a request for calling a function; the server replies with the result of the function;
- **Fire and Forget**: the client sends a request for calling a function, but the server does not reply;
- **Event service**: the client subscribes for a service, then the service sends periodically, or when there is a change, information. (Similar to CAN network);
- **Field service**: with getter and setter methods fields can be read or modified. When a field is changed, a notification is sent.

The four communication methods, which rely on UDP and TCP sockets, are shown in Figure 3.28. To determine if a service is available or not SOME/IP-SD (Service

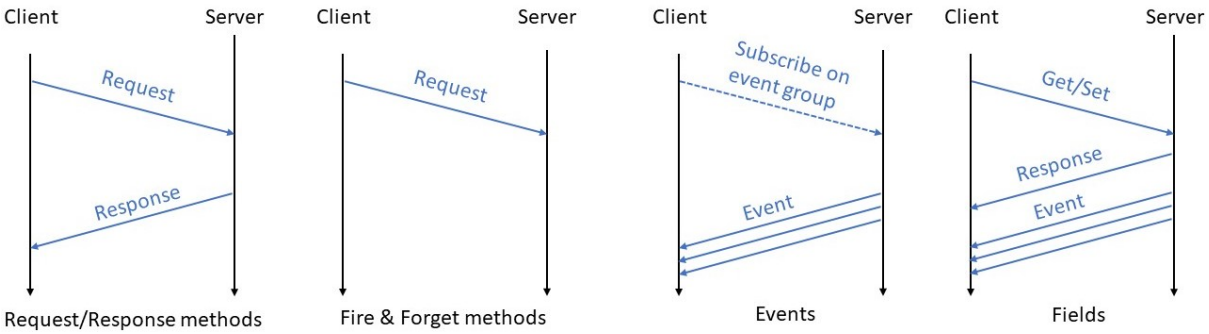

Figure 3.28. SOME/IP communication methods: Request and Response, Fire and Forget, Events, Fields.

Discovery) is used. Especially in start-up phase of the car, Service Discover is very useful for simplifying, reducing the time, since each ECU can communicate its availability. At the same time it is also useful for detecting undetected ECUs that fail and for saving energy by supplying it only to ECUs that need it.
Due to its short initialization time, SOME/IP is suitable for ADAS and infotainment systems, but it cannot be used for strictly real time applications yet, such as motor control.
SOME/IP and SOME/IP-SD headers are shown in Figure 3.29. The fields of SOME/IP header mean: [1]

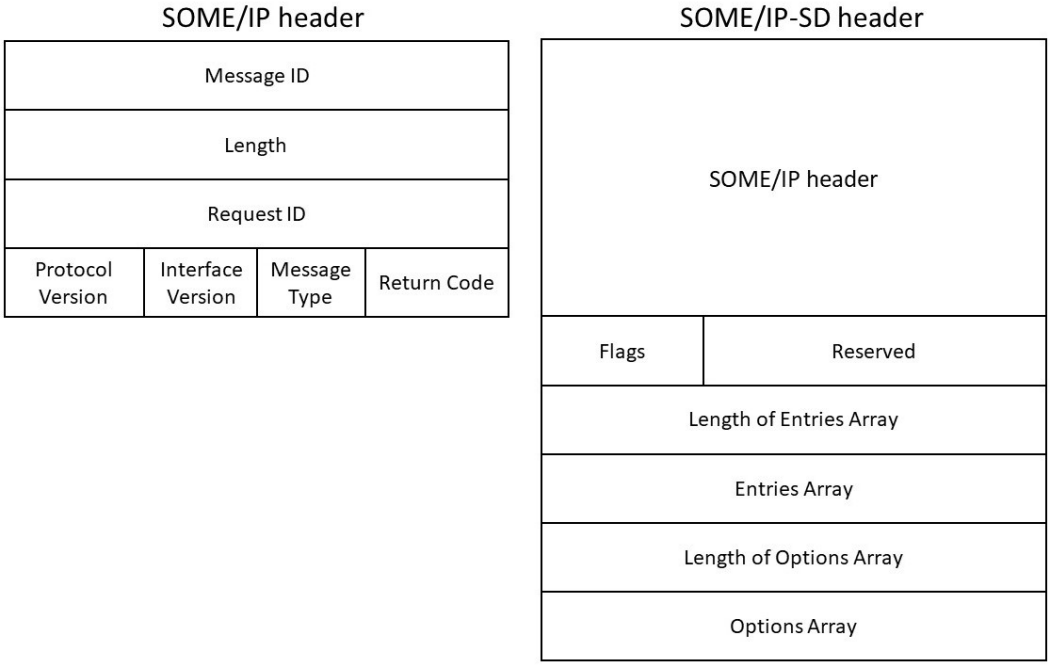

Figure 3.29. SOME/IP and SOME/IP-SD headers.

- **Message ID**: used to identify the Remote Procedure Call to a method of an application or to identify an event;
- **Length**: from the next header field to the end of the payload;
- **Request ID**: to differentiate between multiple calls of the same method;
- **Protocol Version**: SOME/IP version;
- **Interface Version**: Major Version of the Service Interface;
- **Message Type**: used to differentiate different types of messages (e.g. REQUEST, REQUEST_NO_RETURN, RESPONSE, ERROR, ...);

- **Return Code**: to signal to report whether a request has been successfully processed.

SOME/IP uses a decentralized approach where each ECU offers and requests availability to all other ECUs in broadcast mode.

3.7.2 AVB/TSN

Ethernet can not be used for real-time communications or for multimedia purposes because it is not able to guarantee bounded latency, bandwidth allocation, endpoint synchronization. In 2005, IEEE 802.1 created Audio Video Bridging Task Group (AVB Task Group) with the scope of creating a collection of standards to meet all these needs. Firstly, the group focused on audio and video communication, subsequently, in November 2012, changing its name to Time-Sensitive Networking Task Group (TSN Task Group), they studied protocols for safety critical applications that require stringent timing requirements in terms of: low latency, synchronization, no noticeable jitter. At the time of writing many TSN sub-standards are still under development, however the main and most important standards, overview shown in Figure 3.30, will be covered here.

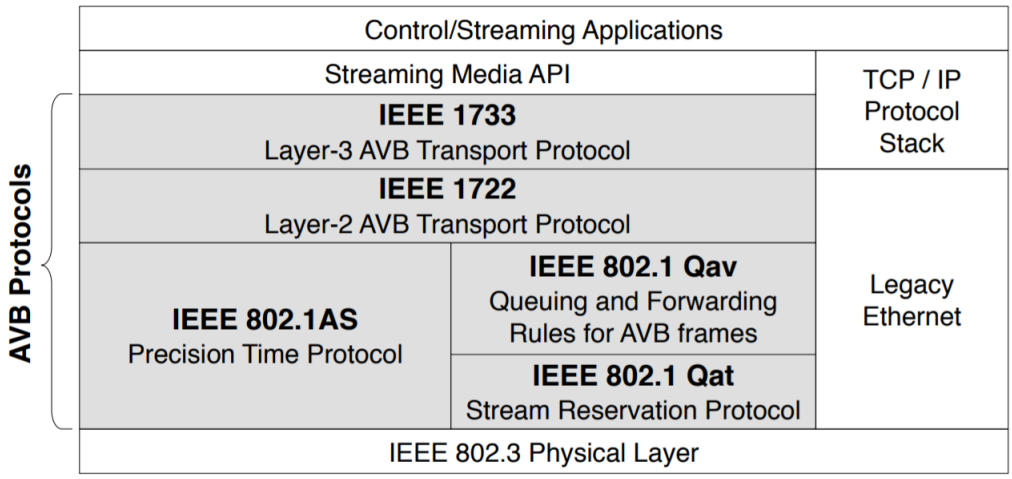

Figure 3.30. Overview of AVB protocols. Source: [57]

As in Figure 3.31, it can be possible to have hybrid networks with some AVB nodes, inside AVB domains, and others not AVB domains, but only AVB-capable end point and switches can transmit or receive AVB traffic.
The first standard, *IEEE 1722-2011* [17], shows how two AVB nodes communicate. Firstly, AVB packet is composed by the following fields:

- **Header**: type of AV data;

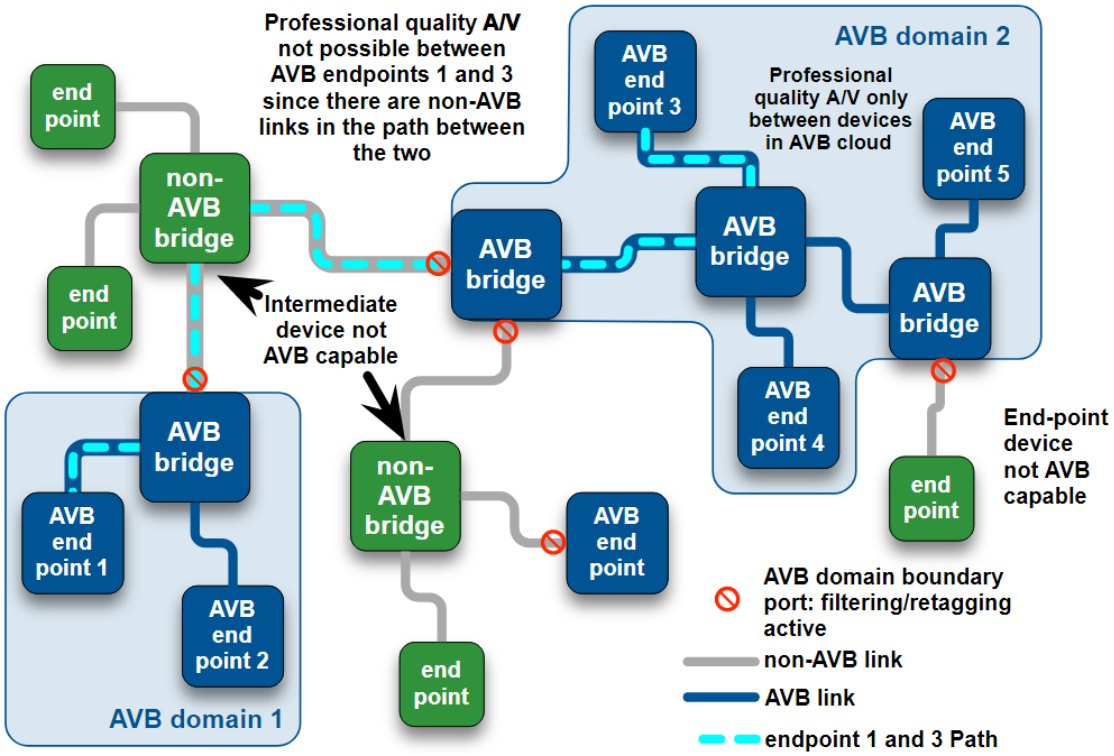

Figure 3.31. Hybrid network with AVB domain and not. Source: [36]

- **Stream ID**: specific data stream derived from MAC address of Talker;
- **Presentation Time**: maximum time when a packet must be received at the Listener;
- **Payload Information**: format of the payload;
- **Payload**.

A packet, shown in Figure 3.32, lies on the first two layers and not on higher layers of OSI model to reduce the processing time and consequently latency. The AVB communication works between talker, source of data, and listener, consumer of data like shown in 3.33. In *IEEE 1722-2016*[19] version more data types are supported. [35]
Synchronization between talker and listener nodes is described in *IEEE 802.1AS-2011*[18] standard with the generalized Precision Time Protocol (gPTP). A node, the Grandmaster, provides all other nodes a synchronization clock as common reference global time with an accuracy of better than $1\mu s$ and precision of some nanoseconds. This node is chosen, dynamically, among those that have the best clock in the AVB network thanks to the Best Master Clock selection Algorithm

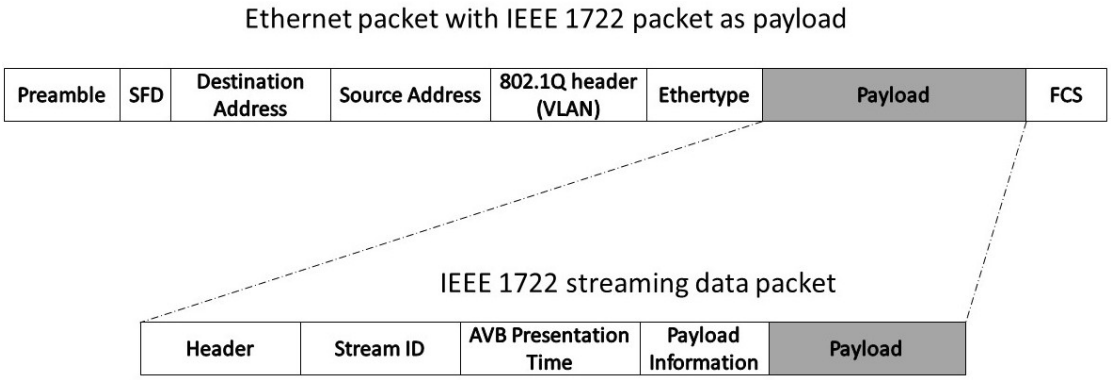

Figure 3.32. IEEE 1722 streaming data packet.

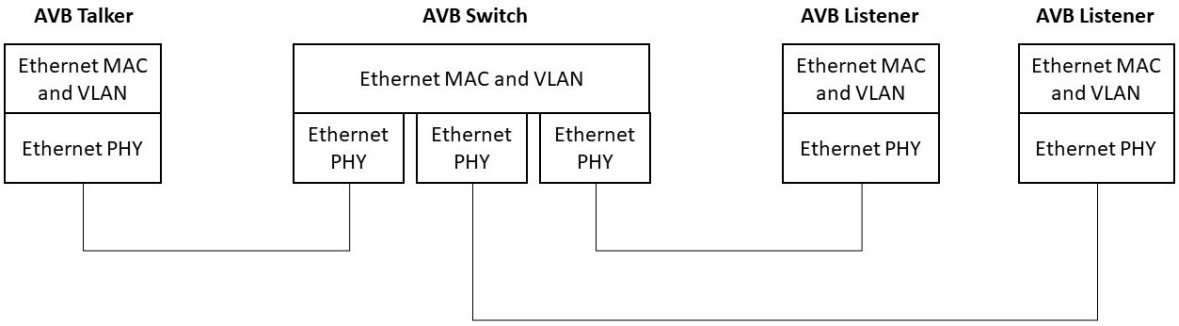

Figure 3.33. AVB topology: Talker and Listener.

(BMCA) where the clock of each node is cyclically checked with the current grandmaster and eventually replaced with the best one. The algorithm also deals with the so called Clock Spanning tree, contains all the paths between AVB nodes, so that the whole network knows the propagation delays (Link Delay) of the different paths. In Figure 3.34 is shown a gPTP domain with a grandmaster. The Link Delay is computed:

$$\frac{(T4 - T1) - (T3 - T2)}{2}$$

as shown in Figure 3.35. Then this delay will be used for clock synchronization between master and slave exchanging Sync and Follow up packets as in Figure 3.36. Typically, the node locations and the link lengths do not change between car ignitions, so delay values are fixed and saved.[35] Since in 2011 version a change of Grandmaster takes too time (more than $200ms$) by BMCA algorithm, with a new standard version: *IEEE 802.1AS-2020*[23], redundant Grandmasters, redundant Clock Spanning trees and multiple gPTP are implemented.
To allocate bandwidth for each application, thanks to *IEEE 802.1Qat*[16] Stream

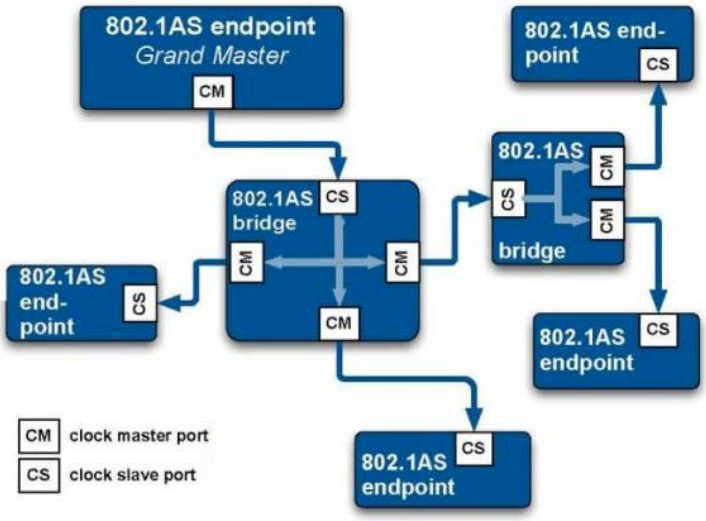

Figure 3.34. gPTP domain. Source: [37]

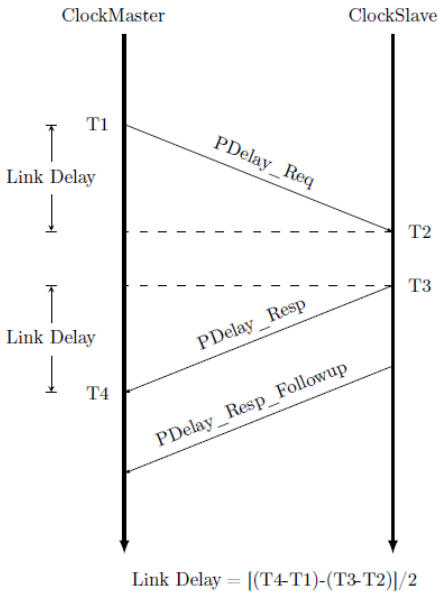

Figure 3.35. Link delay.

Reservation Protocol (SRP), the Talker makes itself available to stream data to all AVB nodes. If a Listener is interested, it communicates it and a reserved stream is established. Certainly it is necessary to guarantee availability of reserved bandwidth, the maximum allowed is 75% for AVB streams (the remaining is used for Best-Effort traffic). This distributed architecture, where all switches have to check

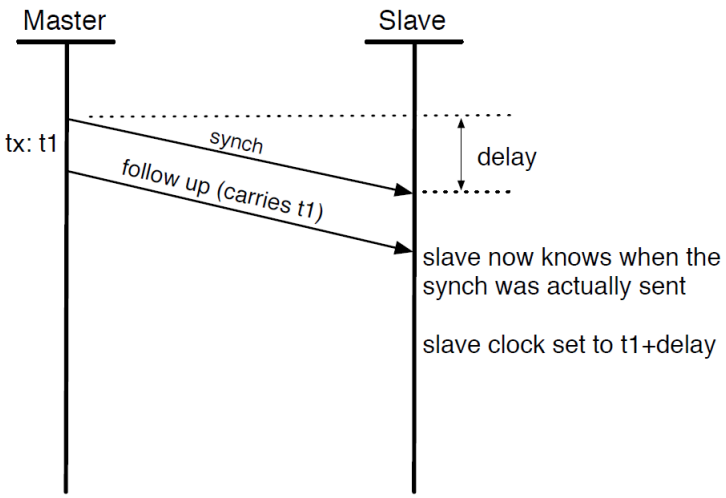

Figure 3.36. Sync and Follow up packets.

individually if the bandwidth is available, is time consuming. So *IEEE 802.1Qcc*[22] specifies a more advanced SRP as centralized network management.
Regarding to traffic, it is possible to define Class A streams (highest priority) and Class B streams or its own user-defined class, as shown in Table 3.2.[35]

Traffic Class	**Transmission Period**	**Expected Latency (7 hops)**
Class A	$125\mu s$	$2ms$
Class B	$250\mu s$	$50ms$

Table 3.2. Priority classes.

IEEE 802.1Qav[15] manages the allocation priority of streams. The Forwarding and Queuing of Time Sensitive Streams (FQTSS) standard takes care of creating message queues and then forwarding them according to the order of priority, splitting between time-critical and non-time-critical traffic, and applying the Credit-Based Shaper (CBS) to avoid traffic overloads and to not block the bus in presence of low priority data as shown in Figure 3.37. Since high jitter provides low quality communication with frame jumps, it is possible to have continuous streams, more uniform, thanks to CBS. In Figure 3.38 an example of AVB traffic shaping is shown: when the credit is greater than or equal to zero the AVB-Queue can start its transmission, in such case the credit will have a negative fixed slope; instead when a frame is waiting in queue the credit will have a positive fixed slope; in case of empty queue the credit is equal to zero. [12]
The scheduling of packets transmission between stations and switches is standardized in *IEEE 802.1Qbv*[21], thanks to Time Aware Shaping (TAS). Since time

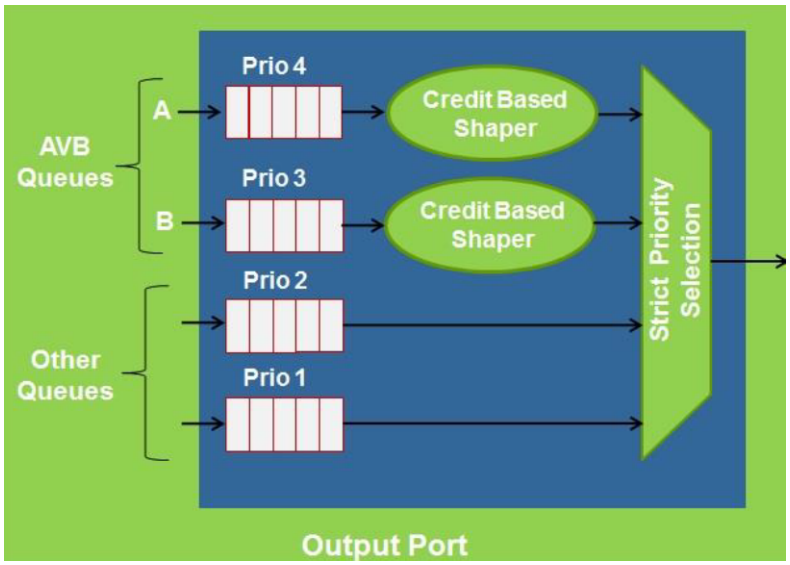

Figure 3.37. FQTSS example. Source: [29]

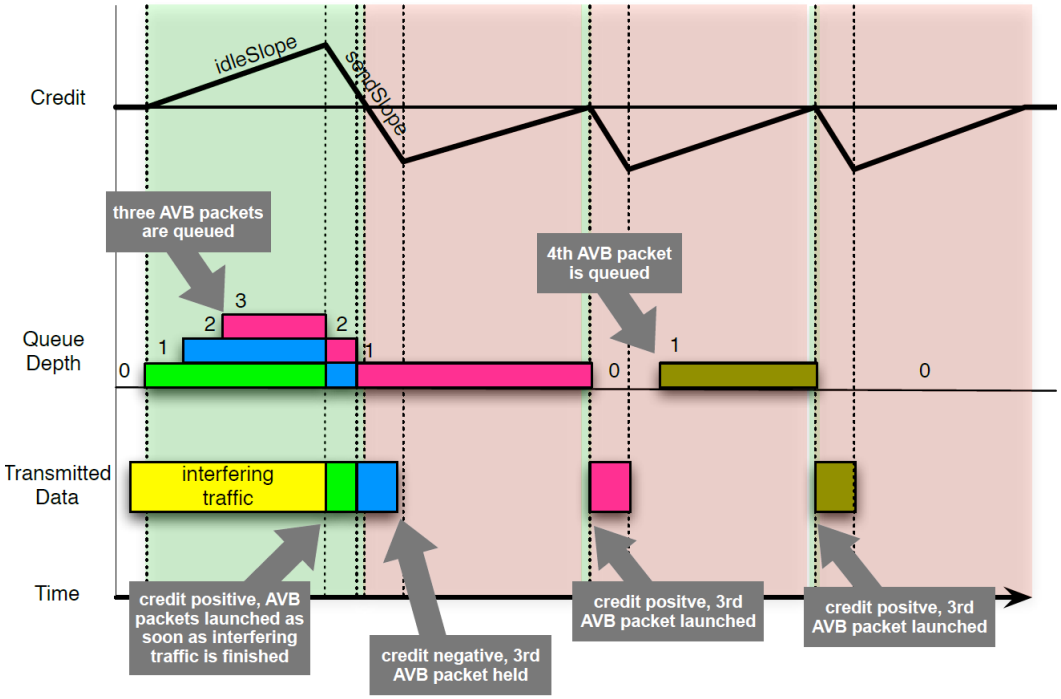

Figure 3.38. AVB traffic-shaping. Source: [38]

division multiple access (TDMA) is implemented if a new frame, too large, arrives in a time slice, as in Figure 3.39, the transmission can not be stopped. TAS with its

guard bands allows not to start new frames, in addition, with *IEEE 802.1Qbu*[20], also preemption is implemented to minimize bandwidth losses due to guard bands (it can not send anything in the guard time). In this case, as in Figure 3.39, the interrupted frame is restored later. The minimum fragment size is 64 bytes. [12]

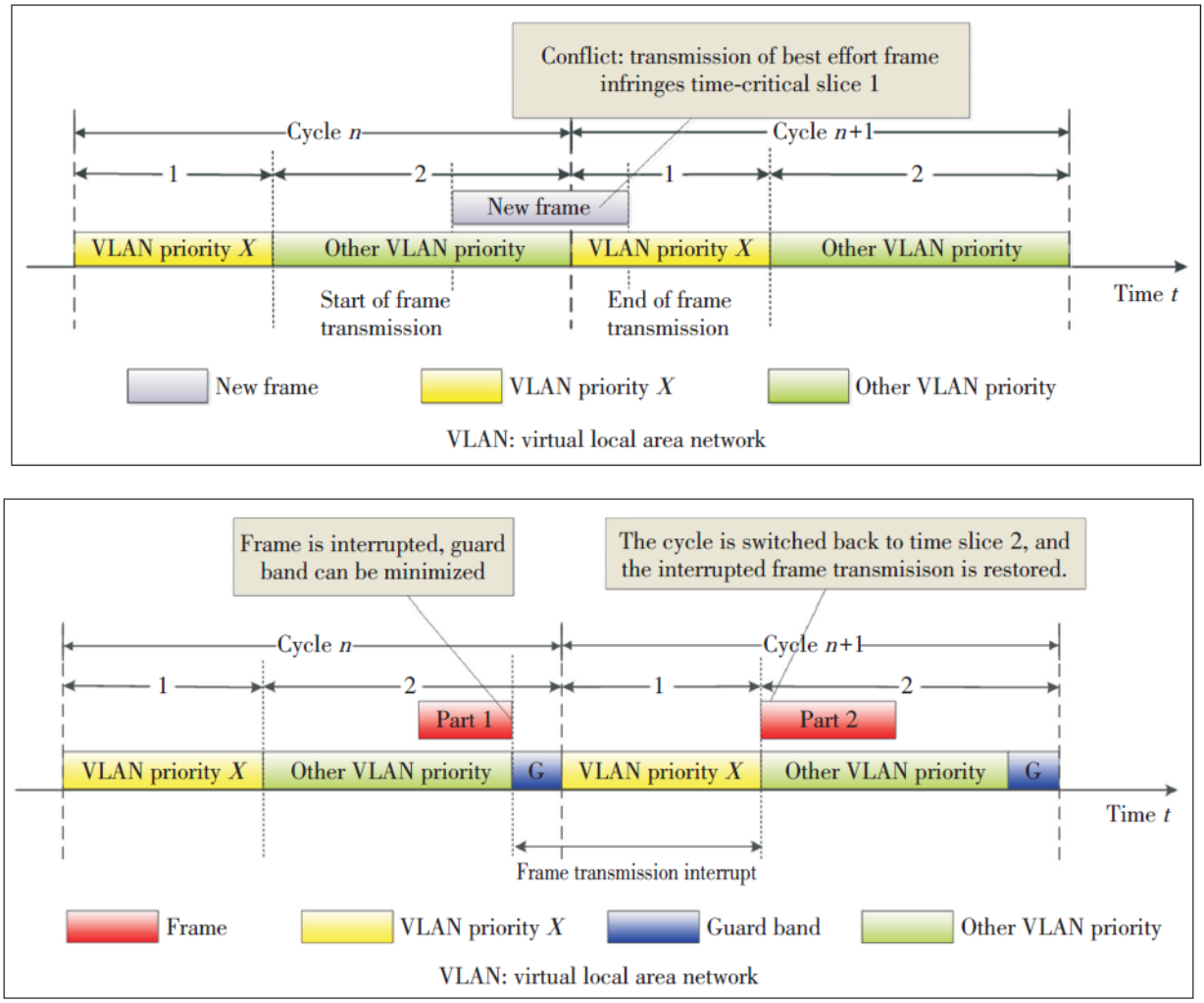

Figure 3.39. Guard bands and frame preemption. Source: [12]

3.7.3 TTEthernet

Time-Triggered Ethernet (TTE or TTEthernet), standardised by the Society of Automotive Engineers (SAE) AS6802 [11] in 2011, extends, remaining compatible, the classical IEEE 802.3 Ethernet to meet requirements of fully deterministic communication with guaranteed constant latency, bandwidth, jitter and fault tolerant

synchronisation to real-time applications.
TTEthernet operates on OSI layer 2, with a frame format compatible with IEEE 802.3 Ethernet, and on many physical layers like 100Base-T1S on the automotive case, it can coexist with other Ethernet network, traffic classes or services, such as AVB, since its backward compatibility. [69][62]
To satisfy different real-time and safety needs, three traffic and message types are provided: [69][2]

- The first class, **time-triggered** (TT), has the precedence over all other traffic classes. Packets are sent at scheduled and predefined times. In this case delays and precisions are certainly guaranteed and predefined. So this class is suitable for brake-by-wire, steer-by-wire and all other stringent real-time systems, specifically in chassis and powertrain domain where vehicle stability and dynamics are crucial from a safety point of view;

- In the second one, **rate-constrained** (RC), real-time requirements are less stringent compared to the previous traffic class. In this case a predefined bandwidth is guaranteed and time requirements have defined upper bounds. This class is suitable for multimedia or safety-critical applications with highly reliable communication, but moderate (not stringent) temporal constraints. In case of stringent requirements TT class must be used;

- In the third one, **best-effort** (BE), the packets are sent in best-effort manner (like in classical Ethernet networks), and therefore there is no guarantee if and when the messages can be transmitted and with what delays. In this case packets are transmitted in a FIFO queue on the remaining bandwidth with lower priority with respect to TT and RC classes. This class is suitable for all legacy IEEE 802.3 Ethernet traffic without any requirement in terms of time (QoS).

In the example of Figure 3.40 a dataflow integration between different frames (TT, RC and BE) is shown. Sender 1 sends TT frames, periodically, with 3ms of period and BE frames. Sender 2, instead, sends TT frames, periodically, with 2 ms of period and both BE and RC frames. At the Switch 1 the TT frames are scheduled with period 6 ms, corresponding to the Least Common Multiple (LCM), instead RC and BE frames are not subject to a predetermined schedule. [58]
There is one last message type, called Protocol Control Frame (PCF), for establishing and maintaining synchronization. These messages, obviously, have the highest priority. Clock synchronization is fundamental for TT frames, for this purpose, sender, switch and receiver always transmit clock synchronization messages in a hierarchical master-slave architecture in a distributed fault-tolerant way, similar to IEEE 1588 Precision Time Protocol previously treated.

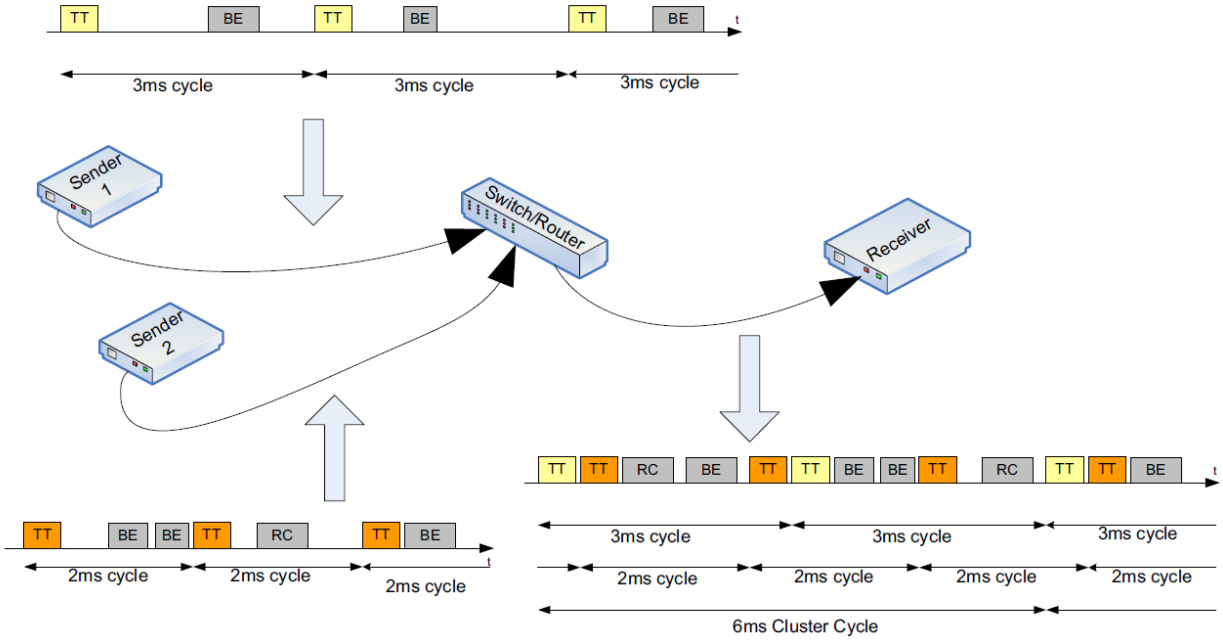

Figure 3.40. Dataflow integration between TT, RC and BE frames. Source: [58]

Comparison between TSN and TTEthernet Both IEEE 802.1 AVB/TSN and Time-triggered Ethernet use real time approach to provide support for time triggered communication. The two protocols have different critical levels and requirements. While the first one allows the synchronous reproduction of the frame and, thanks to its signaling protocol, the dynamic recording of the data flow; the second one, suitable for safety-critical application, owns an offline configured schedule table with better determinism and precision of time in communication. Both use Time-Division-Multiple Access (TDMA).
In general, TSN has more flexibility to adapt configurations modification and bandwidth equity (reserved based on the actual demand) than TTE, on the contrary TSN is only appropriate for soft-real time applications, unlike TTE, because it does not have fault-tolerant clock synchronization and it admits a low-granularity scheduling mechanism. [33]
For an in-depth comparison, please refer to the paper: [71].
In summary, it is advisable to use:

- *AVB/TSN* for multimedia, infotainment, driver assistance and soft real-time applications;

- *TTE* for stringent real-time uses, such as chassis and powertrain applications like vehicle stability, agility and dynamics.

3.7.4 DoIP

Diagnostics over IP (DoIP), standardized on *ISO 13400*[25], has had an important impact in diagnostic area of Automotive Ethernet enabling communication between ECUs and an external tester using IP, TCP and UDP protocols, this to identify and resolve faults, flashing ECUs. AUTOSAR is the software architecture for implementing the communication between ECUs and the external tester. [30]
Instead of Unified Diagnostic Services (UDS) on CAN, that support max data rated of 500kbps, DoIP supports data rates up to 100Mbps, so it is suitable for all applications that require fast data transfer.
DoIP supports on-board and off-board communications. The first one warns the passengers by activating on-board warning, the second one instead stores the fault to be retrieved, subsequently, only in a garage by a mechanic.
An example of vehicle network architecture is shown in Figure 3.41, where a DoIP Gateway is needed to save the effort and cost of integrating DoIP stack in every ECU separately.

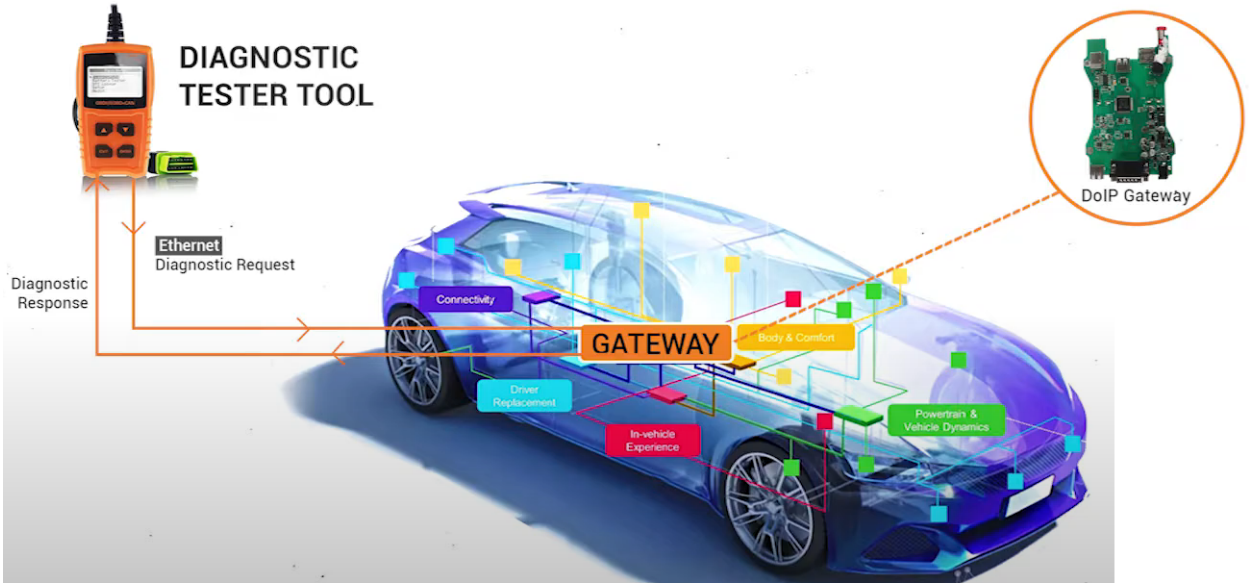

Figure 3.41. DoIP network architecture. Source: [10]

As shown in Figure 3.42, in the communication sequence, the diagnostic tester enables the sending of Diagnostic Request, then the recipient ECU process this request and reply with an acknowledge at the tester. In this process the DoIP Gateway works as intermediary, forwarding requests and responses to the respective ECUs and testers. A DoIP message is shown in Figure 3.43.

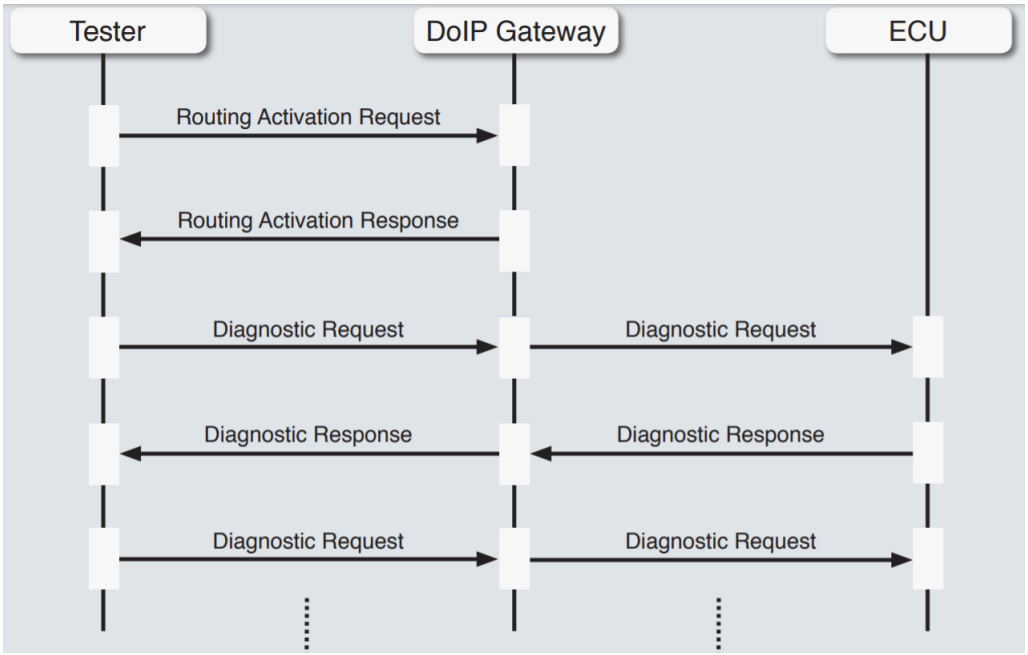

Figure 3.42. Communication sequence. Source: [52]

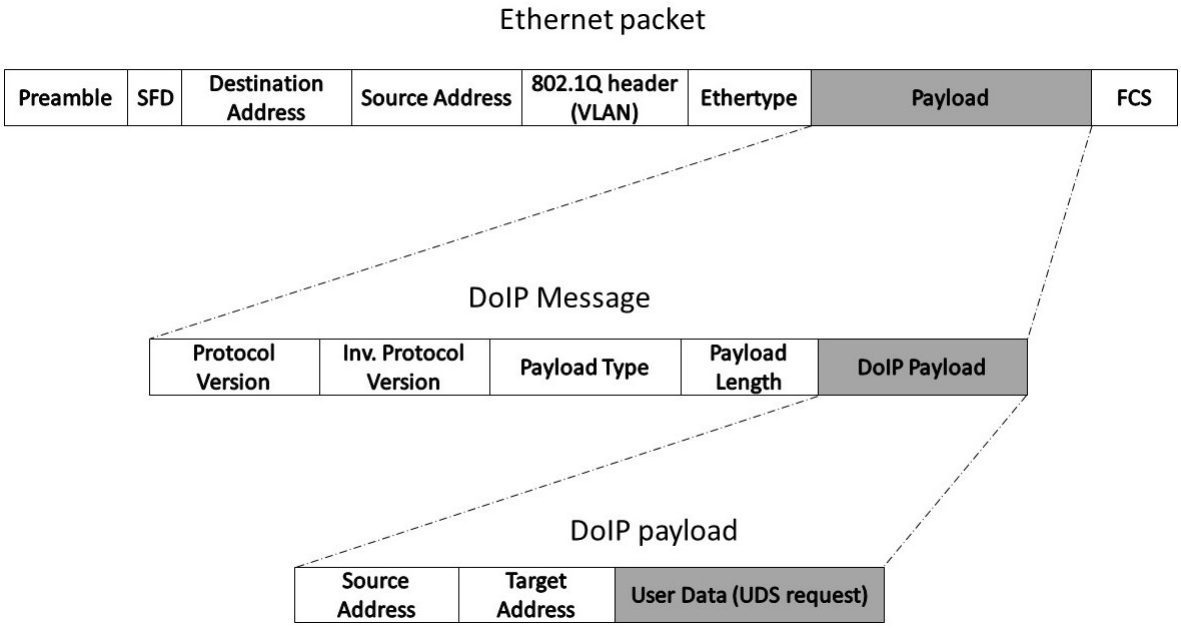

Figure 3.43. DoIP message.

Chapter 4

ECUs on Automotive Ethernet: testing and validation

This chapter describes, according to many automotive companies with its white papers, existing methodologies and strategies for testing and validating interconnected ECUs on Automotive Ethernet.

4.1 Testing and Validation

As mentioned previously in Section 1.2 Automotive Ethernet needs to be tested, the goal is to avoid late discovery of problems, for safety reason, but also to save money and time, in fact it is important not only to find an error, but to find it as soon as possible, according to the automotive V-Cycles, in Figure 1.4, for all the development phase and for all components. It must also be said that testing the TCP/IP stack is much more complex than other protocols such as CAN or FlexRay, however it must be done to increase the quality, reliability and safety of the vehicle. To augment the re-use, quality and reduce the cost, testing has to be under standardization. So OPEN Alliance created standardized tests for its Technical Committee number 8 (TC8) for IPv4, ARP, DHCP, ICMP, IPv4 AUTOCONF, UDP, TCP and SOME/IP.
Following the V-Cycle model on testing and validation phase, firstly, *component-level testing* has to be done to evaluate an ECU as a separate entity with respect to the other ECU, but with accessible communications data. Then, *network-level testing* to evaluate all communication functions and consequently *system-level testing* to examine all the requirement specifications with all ECUs connected. Finally *in-vehicle acceptance test and maintenance* to also solve the cases of error code in malfunctions. [35]
As explained in previous sections Automotive Ethernet, which is often used for safety-critical systems, only has new PHY layers, but everything behind it is pretty

much taken from standard Ethernet with related testing methodologies. However there are new protocols and applications which are to be tested. Since the automotive field, with Automotive Ethernet, has different requirements than local networks, with standard Ethernet, it is good to start testing the network from everything that is already well established for the IT industry. [55]
According to many white papers [55][27][70] tests can be divided into three major types:

- **Conformance and interoperability testing**: studies if a device under test (DUT) works in conformance with the requirements;
- **Negative testing**: studies the response to the system when there are errors or unexpected signals;
- **Performance testing**: studies throughput, latency, frame loss, jitter and various performance parameters.

In particular, at component level, OPEN Alliance has made available Automotive Ethernet ECU Test Specification regarding conformance and interoperability testing [44]. Instead performance testing is usually done according to the Internet Engineering Task Force (IETF) Request for Comments (RFC) 2544, called Benchmarking Methodology for Network Interconnect Devices [51]. In addition, since Automotive Ethernet uses layer 2 switches, RFC 2889 is used to test them [28].

4.2 Conformance Testing

In IT industry, nowadays, all protocols are well standardized and tested, so the compliance and interoperability testing phases are shortened. For the automotive field, on the other hand, it is quite the opposite, because Automotive Ethernet is still new and under development. Consequently, OPEN Alliance has made available Automotive Ethernet ECU Test Specification (TC8) [44] regarding conformance and interoperability testing. In this Test Specifications is checked if a device under test (DUT) works in conformance with the requirements. Open Alliance also suggests the test process, in Figure 4.1, to have a reliable communication between two ECUs starting from the test specifications.
From 2020 the Open Alliance test specifications are divided respecting the OSI level architecture: Layer 1 [41], Layer 2 [42] and from Layer 3 to 7 [43]. In general, a large number of tests for the protocol layers in Figure 4.2, are carried out for each function in an automated way.

Layer 1 For Layer 1, referred to 100Base-T1, tests are divided into:[41]

- *Interoperability*: Link-up time, Signal Quality, Cable diagnostics;

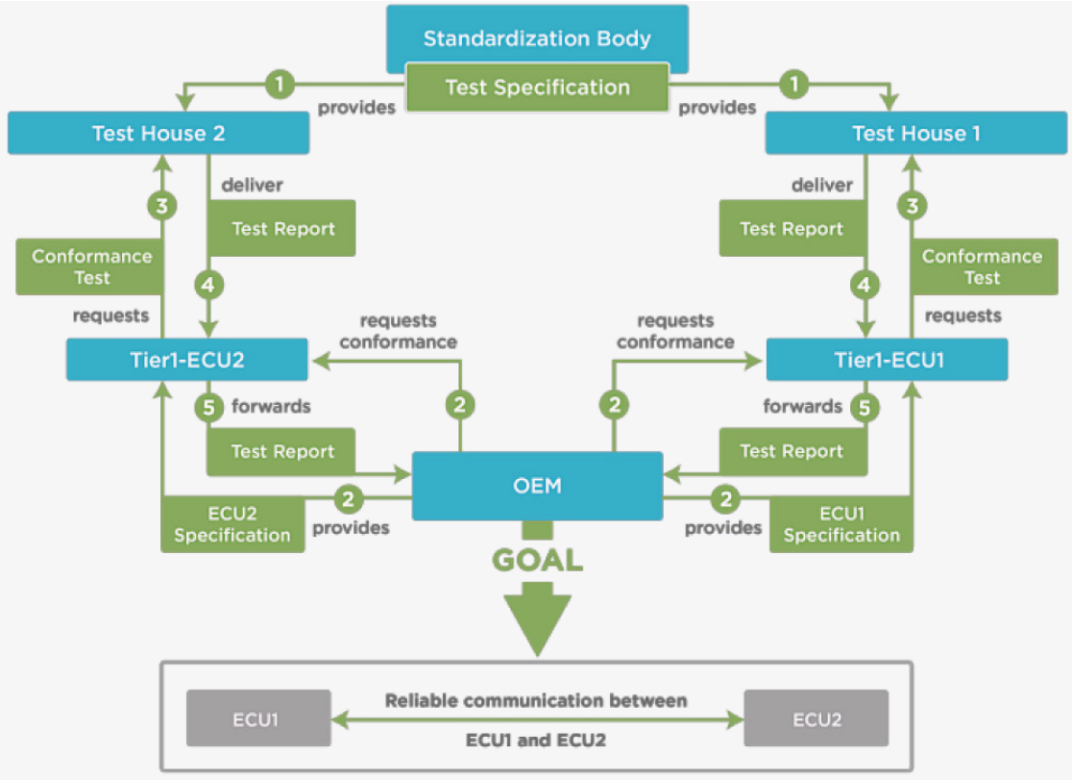

Figure 4.1. Test process for two ECUs starting from test specifications. Source: [44][31]

- *Physical Medium Attachment (PMA) sublayer*: Transmitter Electrical Specifications.

Layer 2 For Layer 2 the tests are [42]: *VLAN Testing, Address Learning, Filtering of incoming frames, Time synchronization, Quality of Service, Configuration.* In Figure 4.3 is shown an example of Address Learning which check if switch supports reading the learned Address Resolution Logic (ARL) table[1]. In this case, but also in all other cases of layer 2 tests is used the standard test setup for switching shown in Figure 4.4.

Layer 3-7 For Layer 3 to 7 the test is divided into: [43]

- *Address Resolution Protocol (ARP)*: Packet Generation, Packet Reception;

[1]Address Resolution Logic (ARL) table: Internal table to the switch containing the correspondence between MAC addresses and ports of the switch)

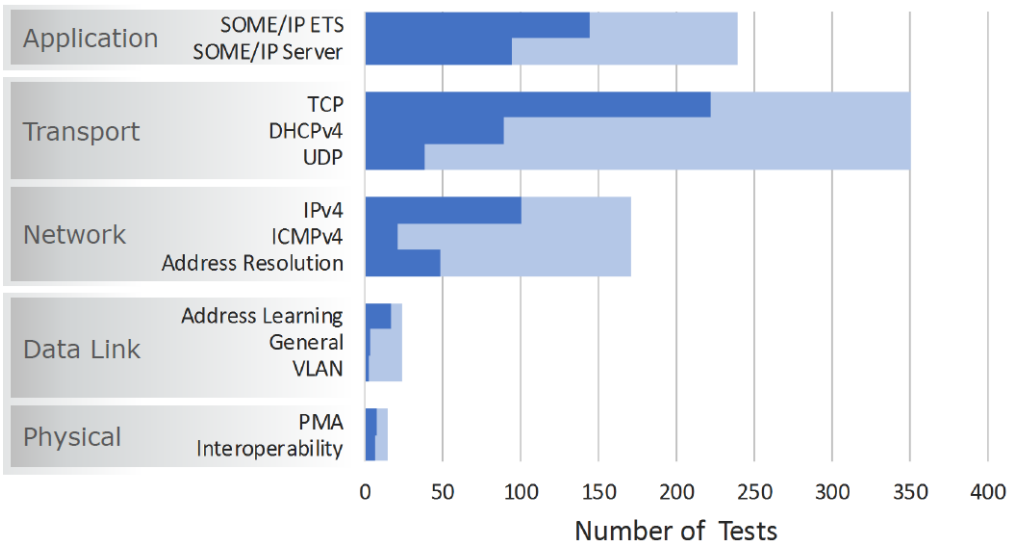

Figure 4.2. Tests number for each test groups. Source: [65]

- *Internet Control Message Protocol Version 4 (ICMPv4)*: Error Handling, ICMP Types;
- *Internet Protocol Version 4 (IPv4)*: IPv4 Header, IPv4 Checksum, IPv4 Time to Live, IPv4 Version Number, IPv4 Addressing, IPv4 Fragments, IPv4 Reassembly;
- *Dynamic configuration of IPv4 Link Local Address*: Introduction, Address Selection, Defense and Delivery, Announcing an Address, Conflict Detection and Defense, Link-Local Packets Are Not Forwarded, Healing of Network Partitions;
- *User Datagram Protocol (UDP)*: UDP Message Format, UDP Datagram Length, UDP Padding, UDP Fields, USER Interface, Introduction, Invalid Addresses;
- *Dynamic Host configuration Protocol Version 4 (DHCPv4) Client*: Summary, The Client-Server Protocol, Client-server interaction – allocating a network address, Client parameters in DHCP, DHCP usage, Constructing and sending DHCP messages, Initialization and allocation of network address, Reacquisition and expiration;
- *Transmission Control Protocol (TCP)*: Connection Establishment and Basic Exercising of the State Machine, Processing and Generating TCP Checksums, Processing Unacceptable Acknowledgements and Out of Window Sequence Numbers, Processing TCP RECEIVE Calls Received from the Application Layer, Processing TCP ABORT Calls Received from the Application Layer,

SWITCH_ADDR_001: Address_Learning_read_ARL_table

Synopsys	Check if switch supports reading the learned ARL table.
Prerequisites	The DUT allows reading the ARL table.
Test setup	Standard test setup for switching
Test Input Parameters	n/a
Test Procedure	1. Through any vendor specified means have the DUT delete its dynamically learned Address Table entries. 2. Wait 10 seconds. 3. From the test station, send at least 2 untagged and at least 2 tagged frames to each of the DUT ports, each frame with a different, valid source MAC address and each of the tagged frames with a different valid VLAN tag according to the VLAN configuration of the corresponding port. 4. Through any vendor specified means, read the learned MAC address table.
Pass Criteria	4. The read table correctly lists the MAC addresses and the corresponding ports as learned in step 3.
Reference	n/a
Notes	The ARL table MAY contain additional addresses that have been learnt from frames that have been generated from applications behind the internal ports.

Figure 4.3. Layer 2 test, called: "SWITCH_ADDR_001: Address_Learning_read_ARL_table". Source: [42]

TCP Packet Flag Generation in Response to Receiving Invalid Packets, Processing TCP Flags, Closing a TCP Connection, Processing of TCP MSS, End of Option List, and NO-Operation Options, Processing Out of Order Segments and Delayed ACKs, Retransmission Timeout, Generation of Zero Window Probes, Nagle Algorithm, Use of the Urgent Pointer, Connection Establishment, Header, Sequence Number, Acknowledgment, Control Flags;

- *Scalable service-Oriented MiddlewarE over IP Protocol (SOME/IP)*: Message Format, Service Discovery Messages, Service Discovery Communication Behaviour, Some/IP Basic Functionality, Specification of the SOME/IP on-wire format, Remote Procedure Call Protocol (RPC) specification, Enhanced Testability Service test cases.

In Figure 4.5 is shown an example of IPv4 Header test case which check if DUT generates an IPv4 Packet with a Total Length greater than or equal to 20. In this case, but also in most other cases of layers 3-7 tests is used the test setup shown in Figure 4.6.
All these tests are subdivided in many subtests. Open Alliance test solutions, obviously, work at *component-level*. Regarding *network-level* tests and validations,

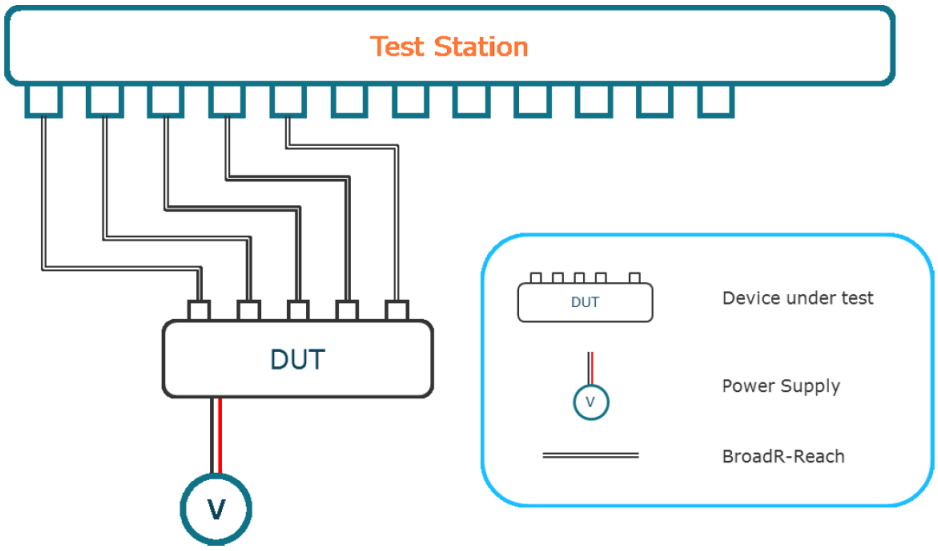

Figure 4.4. Layer 2 Standard test setup for switching. Source: [42]

currently, Open Alliance does not provide anything due to the high complexity and variety of configurations and architectures. Thus there are no general specifications, but individual test solutions are required. [63]

4.3 Performance Testing

As said in IT context Ethernet is well standardized and there are many performance testing methodologies available. Since Automotive Ethernet is quite similar to standard Ethernet, except for PHY layers and some new protocols, it is good to start testing the new networks from everything that is already well known from the IT world, i.e. RFC 2544 and RFC 2889. [53]

4.3.1 RFC 2544

Although with different requirements, more stringent in the automotive field, to test the performance of a network the following parameters are usually considered:

- **Throughput**: amount of data transmitted between two points, measured in bits or packets per second;
- **Latency**: time taken by the data to reach one point to another;
- **Frame loss**: packets lost to reach the destination;

IPv4_HEADER_01: Ensure that the DUT generates an IPv4 Packet with a Total Length greater than or equal to 20.

Synopsis	Ensure that when the DUT is requested to generate an IPv4 packet, then the DUT generates an IPv4 Packet containing an IPv4 Header containing a Total Length indicating a value greater than or equal to 20.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request. 2. DUT: Generates an ICMPv4 Echo Reply including a Total Length Frame in the IPv4 Header greater or equal to 20
Pass Criteria	2. DUT: Generates an ICMPv4 Echo Reply including a Total Length Frame in the IPv4 Header greater or equal to 20
Test Iterations	
Reference	Derived from RFC791, section 3.1
Notes	

Figure 4.5. Layers 3-7 test, called: "IPv4_HEADER_01: Ensure that the DUT generates an IPv4 Packet with a Total Length greater than or equal to 20". Source: [43]

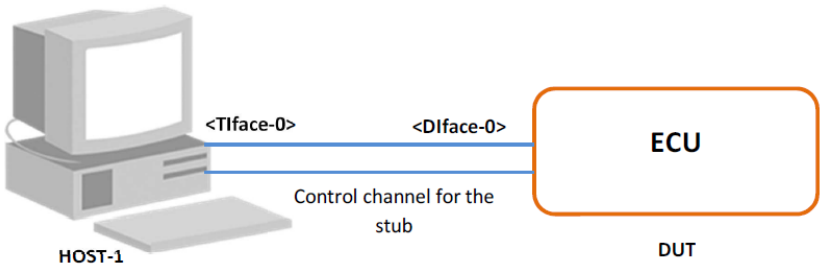

Figure 4.6. Layers 3-7 Simulated Topology. Source: [43]

- **Back-to-back Frames**: number of frames in the longest burst of frames, at the highest throughput, the node can support without frame loss.

In 1999 Internet Engineering Task Force (IETF) created the Request for Comments (RFC) 2544, called Benchmarking Methodology for Network Interconnect Devices

[51] defining, univocally for all vendors, a set of tests to measure the performance of an Ethernet network. The test setups used for all tests, with separate sender and receiver or a tester which supports both functions, are shown in Figure 4.7.

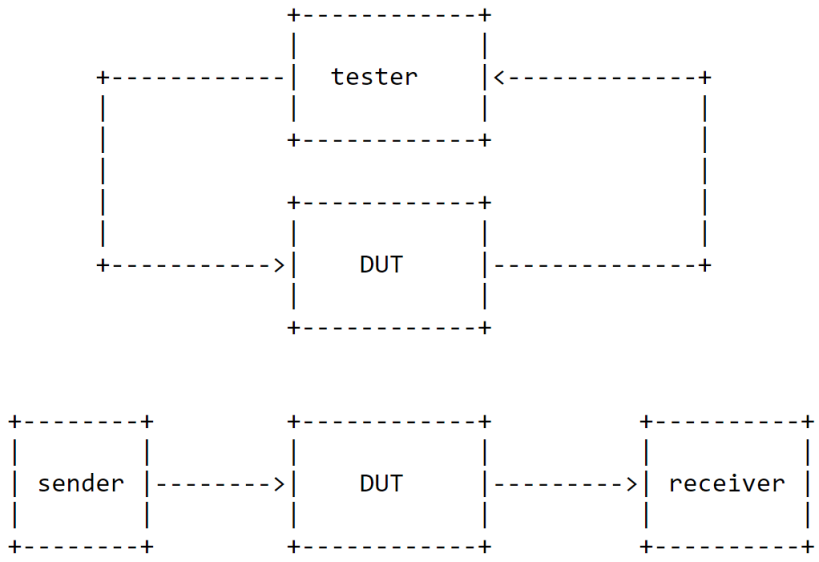

Figure 4.7. RFC2544 test setup. Source: [51]

All the following tests must be performed using a number of different frame sizes, i.e. 64, 128, 256, 512, 1024, 1280, 1518.

Throughput test Throughput test is used to test if the network is capable of carrying traffic at the increasing specified transmission rate, without frame losses. Firstly, driver side requests at receiver side MAC learning frames. This will be sent through frame IP from RFC2544 module to CPU. Then, the driver informs the receiver that a test is ready and waits for an acknowledgment.
The transmission rate used, at this moment, is:

$$R_0 = CIR + EIR$$

where CIR stands for Committed Information Rate and means guaranteed minimum bandwidth (represented in green colour), instead EIR stands for Excess Information Rate and means extra bandwidth (represented in yellow colour).
After the sending of all throughput test traffic, with a specific frame size, the receiver reports:

– If test is passed or failed;

– Number of green and yellow frames received;

– For each frame: size, rate, duration;

– Total elapsed time.

Finally, if the number of received frames (yellow and green) is equal to the number of generated frames, the step test is passed. If a step is not passed or if there is an error, the transmission rate is reduced by RFC2544 RateStep at:

$$R_i = R_{i-1} - R_0 * RateStep/100$$

The process ends when the step rate Ri is greater than 10% of $R0$ or when two consecutive steps pass.
In fact, to pass an entire test, for a specific frame size, two consecutive step tests must have passed; instead to fail an entire test at least one of the two step tests must have failed.[51][61]
An example of throughput measurement is shown in Figure 4.8. Instead in Figure 4.9 is shown an example of throughput test results.

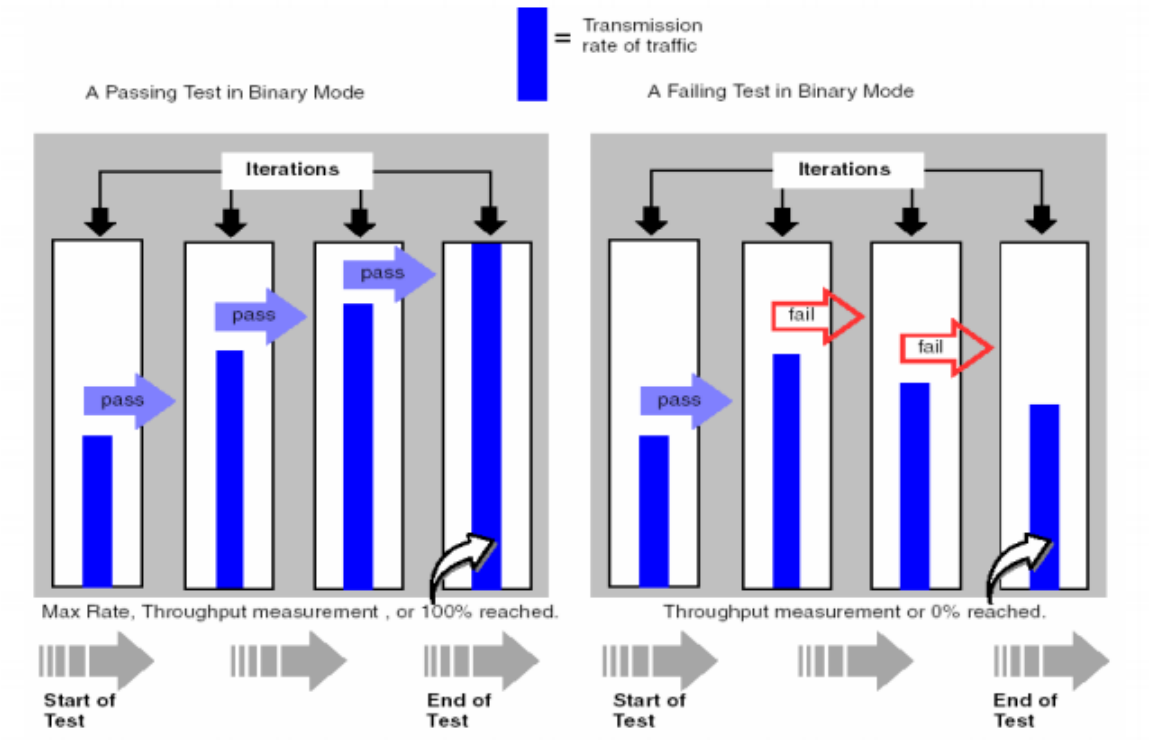

Figure 4.8. Iterations in the Throughput measurement. Source: [34]

Latency test Latency tests are executed at the same time and same number of steps as throughput tests.

Status	Pass
Elapsed Time	217210 ms
Step Length	10000 ms

Step	Direction	Frame Size	Actual Tx Rate	Test Step Duration	Tx Frames	Rx Green Frames	Rx Yellow Frames	Test Step Result
1	NE->FE	64 bytes	1000000 bps	14020 ms	18382	18382	0	pass
2	NE->FE	64 bytes	1000000 bps	14020 ms	18382	18382	0	pass
3	NE->FE	128 bytes	1000000 bps	14020 ms	9469	9469	0	pass
4	NE->FE	128 bytes	1000000 bps	14020 ms	9469	9469	0	pass
5	NE->FE	256 bytes	1000000 bps	13000 ms	4807	4807	0	pass
6	NE->FE	256 bytes	1000000 bps	14020 ms	4807	4807	0	pass
7	NE->FE	512 bytes	1000000 bps	14020 ms	2422	2422	0	pass
8	NE->FE	512 bytes	1000000 bps	13000 ms	2422	2422	0	pass
9	NE->FE	1024 bytes	1000000 bps	13000 ms	1215	1215	0	pass
10	NE->FE	1024 bytes	1000000 bps	13000 ms	1215	1215	0	pass
11	NE->FE	1280 bytes	1000000 bps	13000 ms	973	973	0	pass
12	NE->FE	1280 bytes	1000000 bps	14030 ms	973	973	0	pass
13	NE->FE	1518 bytes	999999 bps	14020 ms	821	821	0	pass
14	NE->FE	1518 bytes	999999 bps	14020 ms	821	821	0	pass
15	NE->FE	9000 bytes	1000000 bps	13010 ms	138	138	0	pass
16	NE->FE	9000 bytes	1000000 bps	13010 ms	138	138	0	pass

Figure 4.9. Throughput test results. Source: [61]

Latency value is computed as the time at which a specific frame is received minus the time at which a frame is fully transmitted.

$$Latency = Received\_Timestamp – Sent\_Timestamp$$

The RFC says that the test must be repeated at least 20 times and an average is taken into consideration.
The test contains:

- If test is passed or failed (Traffic Loss or not);
- Number of green and yellow frames received;
- For each frame: size, rate, duration;
- Total elapsed time.

In Figure 4.10 is shown an example of latency test results.

Frame Loss test Frame loss test is used to test if the network is capable of carrying traffic at the increasing specified transmission rate, with limited and acceptable green CIR frame losses and without out of sequence frames.
As for the throughput tests, driver side requests at receiver side MAC learning frames. Then, the driver informs the receiver that a test is ready and waits for an acknowledgment.
Also for frame loss test, the transmission rate used, at this moment, is:

$$R_0 = CIR(green) + EIR(yellow)$$

Status	Pass
Elapsed Time	217210 ms
Step Length	10000 ms

Step	Direction	Frame Size	Actual Tx Rate	Test Step Result
1	NE->FE	64 bytes	1000000 bps	pass
2	NE->FE	64 bytes	1000000 bps	pass
3	NE->FE	128 bytes	1000000 bps	pass
4	NE->FE	128 bytes	1000000 bps	pass
5	NE->FE	256 bytes	1000000 bps	pass
6	NE->FE	256 bytes	1000000 bps	pass
7	NE->FE	512 bytes	1000000 bps	pass
8	NE->FE	512 bytes	1000000 bps	pass
9	NE->FE	1024 bytes	1000000 bps	pass
10	NE->FE	1024 bytes	1000000 bps	pass
11	NE->FE	1280 bytes	1000000 bps	pass
12	NE->FE	1280 bytes	1000000 bps	pass
13	NE->FE	1518 bytes	999999 bps	pass
14	NE->FE	1518 bytes	999999 bps	pass
15	NE->FE	9000 bytes	1000000 bps	pass
16	NE->FE	9000 bytes	1000000 bps	pass

Figure 4.10. Latency test results. Source: [61]

After the sending of all throughput test traffic, with a specific frame size , the receiver reports:

- If test is passed or failed;
- Number of green and yellow frames received;
- Frame loss ratio;
- Out-of-sequence events;
- For each frame: size, rate, duration;
- Total elapsed time.

The test is passed if Frame Loss Rate (FLR), computed as

$$FLR = (TxFrames - RxGreenFrames)/TxFrames$$

is less than a defined Frame Loss Ratio during the first two consecutive test steps AND Out of Sequence counter is equal to 0.
Also for this test if a step is not passed or if there was an error, the transmission rate is reduced by RFC2544 RateStep at:

$$R_i = R_{i-1} – R_0 * RateStep/100$$

The process ends when the step rate Ri is greater than 10% of R_0 or when two consecutive steps pass.
In fact, to pass an entire test, for a specific frame size, two consecutive step tests must have passed; instead to fail an entire test at least one of the two step tests must have failed. [51][61]
In Figure 4.11 is shown an example of frame loss test results.

Status	Pass
Elapsed Time	217210 ms
Step Length	10000 ms

Step	Direction	Frame Size	Actual Tx Rate	Test Step Duration	Tx Frames	Rx Green Frames	Rx Yellow Frames	Frame Loss Ratio	Out-of-Sequence Events	Test Step Result
1	NE->FE	64 bytes	1000000 bps	14020 ms	18382	18382	0	0.00	0	pass
2	NE->FE	64 bytes	1000000 bps	14020 ms	18382	18382	0	0.00	0	pass
3	NE->FE	128 bytes	1000000 bps	14020 ms	9469	9469	0	0.00	0	pass
4	NE->FE	128 bytes	1000000 bps	14020 ms	9469	9469	0	0.00	0	pass
5	NE->FE	256 bytes	1000000 bps	13000 ms	4807	4807	0	0.00	0	pass
6	NE->FE	256 bytes	1000000 bps	14020 ms	4807	4807	0	0.00	0	pass
7	NE->FE	512 bytes	1000000 bps	14020 ms	2422	2422	0	0.00	0	pass
8	NE->FE	512 bytes	1000000 bps	13000 ms	2422	2422	0	0.00	0	pass
9	NE->FE	1024 bytes	1000000 bps	13000 ms	1215	1215	0	0.00	0	pass
10	NE->FE	1024 bytes	1000000 bps	13000 ms	1215	1215	0	0.00	0	pass
11	NE->FE	1280 bytes	1000000 bps	13000 ms	973	973	0	0.00	0	pass
12	NE->FE	1280 bytes	1000000 bps	14030 ms	973	973	0	0.00	0	pass
13	NE->FE	1518 bytes	999999 bps	14020 ms	821	821	0	0.00	0	pass
14	NE->FE	1518 bytes	999999 bps	14020 ms	821	821	0	0.00	0	pass
15	NE->FE	9000 bytes	1000000 bps	13010 ms	138	138	0	0.00	0	pass
16	NE->FE	9000 bytes	1000000 bps	13010 ms	138	138	0	0.00	0	pass

Figure 4.11. Frame loss test results. Source: [61]

Back-to-back Frames test This test is used to test if a network supports burst transmission up to a specified limit, helping to determine the node buffer capacity at the highest possible speed, measuring the longest burst where there are no lost packages.
Back-to-back frames is executed after all other tests. As for throughput and frame loss tests, driver side requests at receiver side MAC learning frames. Then, the driver informs the receiver that a test is ready and waits for an acknowledgment. The traffic is tested at CIR rate with, initially,

$$BurstSize = CommittedBurstSize(CBS) + ExcessBurstSize(EBS)$$

The driver sends, at defined CBS rate, as many frames as possible of a specified size, in such a way that leaking buckets become nearly empty. This value is:

$$BucketFill = min \begin{cases} 0.99 * BurstSize \\ BurstSize - 1.5 * FrameSize \end{cases}$$

The number of frame required is

$$\sum_{i=0}^{N} FrameSize(i) < BucketFill * \left(1 + \frac{CIR}{LineRate - CIR}\right)$$

which means a large burst to empty the $CBS + EBS$ leaky bucket.
Current cycle is passed if FLR is less than an acceptable ratio, where

$$FLR = (TxFrames - RxGreenFrames)/TxFrames$$

If current cycle is failed a decreased new

$$BurstSize_i = BurstSize_{i-1} – BurstSize_0 * RateStep/100$$

is used. [61]
The test contains:

– If test is passed or failed;

– Number of green and yellow frames received;

– Frame loss ratio;

– For each Burst step: size, rate, duration;

– Total elapsed time.

In Figure 4.12 is shown an example of Back-to-back test results.

Status	Fail NE
Elapsed Time	618010 ms
Step Length	10000 ms

Step	Direction	Frame Size	Burst Size	Test Step Duration	Tx Burst	Rx Green Frames	Rx Yellow Frames	Frame Loss Ratio	Test Step Result
1	NE->FE	64 bytes	64000 bytes	11280 ms	19315	18539	0	4.02	fail
2	NE->FE	64 bytes	44800 bytes	11280 ms	19036	18539	0	2.61	fail
3	NE->FE	64 bytes	25600 bytes	11280 ms	18756	18538	0	1.16	fail
4	NE->FE	64 bytes	6400 bytes	11280 ms	18475	18475	0	0.00	pass
5	NE->FE	64 bytes	6400 bytes	11280 ms	18475	18475	0	0.00	pass
6	NE->FE	128 bytes	64000 bytes	11270 ms	9950	9561	0	3.91	fail
7	NE->FE	128 bytes	44800 bytes	11280 ms	9806	9561	0	2.50	fail
8	NE->FE	128 bytes	25600 bytes	11290 ms	9662	9561	0	1.05	fail
9	NE->FE	128 bytes	6400 bytes	11280 ms	9517	9517	0	0.00	pass
10	NE->FE	128 bytes	6400 bytes	11280 ms	9517	9517	0	0.00	pass
11	NE->FE	256 bytes	64000 bytes	11280 ms	5052	4857	0	3.86	fail
12	NE->FE	256 bytes	44800 bytes	11280 ms	4978	4857	0	2.43	fail
13	NE->FE	256 bytes	25600 bytes	11270 ms	4905	4856	0	1.00	fail
14	NE->FE	256 bytes	6400 bytes	11270 ms	4831	4831	0	0.00	pass
15	NE->FE	256 bytes	6400 bytes	11270 ms	4831	4831	0	0.00	pass
16	NE->FE	512 bytes	64000 bytes	11270 ms	2545	2448	0	3.81	fail
17	NE->FE	512 bytes	44800 bytes	11270 ms	2508	2448	0	2.39	fail
18	NE->FE	512 bytes	25600 bytes	11270 ms	2471	2448	0	0.93	fail
19	NE->FE	512 bytes	6400 bytes	11270 ms	2433	2433	0	0.00	pass
20	NE->FE	512 bytes	6400 bytes	11270 ms	2433	2433	0	0.00	pass
21	NE->FE	1024 bytes	64000 bytes	10250 ms	1276	1228	0	3.76	fail
22	NE->FE	1024 bytes	44800 bytes	10250 ms	1258	1228	0	2.38	fail
23	NE->FE	1024 bytes	25600 bytes	10250 ms	1239	1228	0	0.89	fail
24	NE->FE	1024 bytes	6400 bytes	10250 ms	1220	1220	0	0.00	pass
25	NE->FE	1024 bytes	6400 bytes	10250 ms	1220	1220	0	0.00	pass
26	NE->FE	1280 bytes	64000 bytes	11270 ms	1022	983	0	3.82	fail
27	NE->FE	1280 bytes	44800 bytes	11270 ms	1007	983	0	2.38	fail
28	NE->FE	1280 bytes	25600 bytes	11270 ms	992	983	0	0.91	fail
29	NE->FE	1280 bytes	6400 bytes	11270 ms	977	977	0	0.00	pass
30	NE->FE	1280 bytes	6400 bytes	11270 ms	977	977	0	0.00	pass
31	NE->FE	1518 bytes	64000 bytes	11270 ms	862	829	0	3.83	fail
32	NE->FE	1518 bytes	44800 bytes	11270 ms	849	829	0	2.36	fail
33	NE->FE	1518 bytes	25600 bytes	11270 ms	837	829	0	0.96	fail
34	NE->FE	1518 bytes	6400 bytes	11270 ms	824	824	0	0.00	pass
35	NE->FE	1518 bytes	6400 bytes	11270 ms	824	824	0	0.00	pass
36	NE->FE	9000 bytes	64000 bytes	10250 ms	144	138	0	4.17	fail
37	NE->FE	9000 bytes	44800 bytes	10250 ms	142	138	0	2.82	fail
38	NE->FE	9000 bytes	25600 bytes	10250 ms	140	138	0	1.43	fail
39	NE->FE	9000 bytes	6400 bytes	10250 ms	139	138	0	0.72	fail

Figure 4.12. Back-to-back Frames test results. Source: [61]

4.3.2 RFC 2889

Since Automotive Ethernet uses layer 2 switches, RFC 2889, created in 2000, it can be used, extending RFC 2544, to test them.

RFC 2889 is called LAN Switch Benchmarking Methodology and, extending RFC 2544, provides methodologies to benchmark switching devices, congestion control, latency, forwarding performance, address handling and address filtering.
Using the general test setup described above in RFC 2544, it contains, for different port traffic patterns, traffic loads and frame size, the following test types: [28][54][26]

- *Fully meshed throughput, frame loss and forwarding rates*: To test if the switch can handle fully meshed traffic for different traffic loads. Fully meshed means from all ports to all ports. In fact the test result shows how many frames are transmitted from all the ports and how many are received on all the ports with the percentage of lost frames, so forwarding rate and throughput;
- *Partially meshed one-to-many/many-to-one*: Instead of sending traffic from all ports to all ports, it is sent from one-to-many ports or many-to-one port. In this way it can be determined the maximum rate of reception and forwarding. Test result shows forwarding rate for each frame size;
- *Partially meshed multiple devices*: In this test two switches under test are connected in series to test if they can handle traffic between all ports of multiple devices. Test result shows throughput and forwarding rate for each frame size;
- *Partially meshed unidirectional traffic*: This test is used to show how the switch handles unidirectionally the traffic between one half test ports to the other half. Unidirectionally means that the transmit ports do not receive frames and the receive ports do not transmit them. Test result shows throughput and forwarding rate for each frame size;
- *Congestion control*: To test if and how the switch handles congestion control mechanism, for example if the congested port also affects the non-congested port. The test provides frame loss percentage and forwarding rate for each frame size;
- *Forward pressure and maximum forwarding rate*: The tests are divided in two parts. The first part overloads the switch and measure the forward pressure, sending traffic at interframe gap (time pause between packets) of 88 bits (IEEE 802.3 standard permits more than 96 bits). If the switch transmits the traffic at less than 96bits, forward pressure is detected. The second part shows the maximum forwarding rate as the highest forwarding rate;
- *Address caching capacity*: The test uses binary search with the purpose of establishing the address table (between MAC address and switch ports) size of the switch;
- *Address learning rate*: The test provides the address learning rate transmitting frames with multiple address;

- *Errored frames filtering*: Test if the switch filters frames with some errors, such as CRC, oversize, alignment, undersize, dribble bit and so on;
- *Broadcast frame forwarding and latency*: The test shows the maximum rate to which broadcast frames are received and forwarded. So throughput and latency for each frame size and load.

Chapter 5

Simulations

This chapter is very useful to understand how the Automotive Ethernet stack and the related real-time protocols (i.e. AVB and TTEthernet) work through simulations. All the simulation environment and the related frameworks used are open-source.

5.1 Simulation environment

For simulating Automotive Ethernet will be used: (Objective Modular Network Testbed in C++) OMNeT++, which is a simulation library, INET, a framework supporting communication networks and (Communication over Real-time Ethernet for INET) Core4INET which is an extension of INET for real-time Ethernet on Automotive field.

5.1.1 OMNeT++

OMNeT++[45] is a discrete event network simulation framework, used in many applications, such as: queueing networks, communication networks, multiprocessors systems, performance evaluation and so on. Although OMNeT ++ is often known as a network simulator, in reality, by itself, it is not because it includes the infrastructure and tools to simulate, but not the components to model, specifically, computer, ECU, queueing networks. For this purpose INET framework and Core4INET will be used. [46]
A model in OMNeT++ is composed by: simple modules, compound modules, connections and parameters.
In network simulations, *simple modules* are implemented in C++ and represent sources and sinks, for example protocol entities, such as TCP, routing tables and so on. *Compound modules*, instead, are more simple modules together or, hierarchically, many other compound modules, such as hosts or routers. Connections link

input, output and in/out gates to exchange messages. Finally *parameters* are used to configure data of the modules.
An OMNeT++ project consist of: its (Domain Specific Language) DSL in a *.ned* (Network Topology Description) file containing the architecture of the network, viewable both textually in the editor and graphically in the IDE; *.ini* file containing simulation settings and parameters.
After simulating, results are created in a folder where scalar and vector data can be extrapolated and then plotted graphically.

5.1.2 INET & CoRE4INET

INET [47] is an open-source framework contains many models for all protocols (Ethernet, TCP, UDP, IP, ...) according to all OSI layers. For design and validate new networks or protocols INET is useful, in fact many research groups, like CoRE, they took it as a basis and extended. [68]
CoRE[6] is a research group of Hamburg University of Applied Sciences (HAW-Hamburg) founded by Prof. Dr. Franz Korf and Prof. Dr. Thomas C. Schmidt. This research group created (Communication over Real-Time Ethernet for INET) CoRE4INET (GitHub:[7] and paper: [56]), which is an extension of INET, that supports many automotive real-time protocols like those shown in this thesis (i.e. AVB/TSN, AS6802 TTE).
Since NED language, used usually in OMNeT++, is a bit complex and long, CoRE4inet also supports an Abstract Network Description Language (ANDL) as plug-in. The advantage is that it is possible to write less code lines respect to NED file and then the compilation from ANDL to NED and INI files will be automatic by Eclipse Xtext[1].
An overview of the simulation environment is shown in Figure 5.1.

5.2 Automotive Ethernet simulation

The objective of simulations is to understand how automotive real-time (AVB, TTE) and not real-time messages (Best-Effort) transit at the same time, while finding the time limits under overload conditions.
Using the aforementioned environment and related frameworks, it was chosen to simulate:

– Ethernet AVB (with one class A message and one class B message), usually used for multimedia, driver assistance systems and all applications which require latency and jitter in the order of some millisecond;

[1]Xtext: open-source framework for developing programming and domain-specific languages.[8]

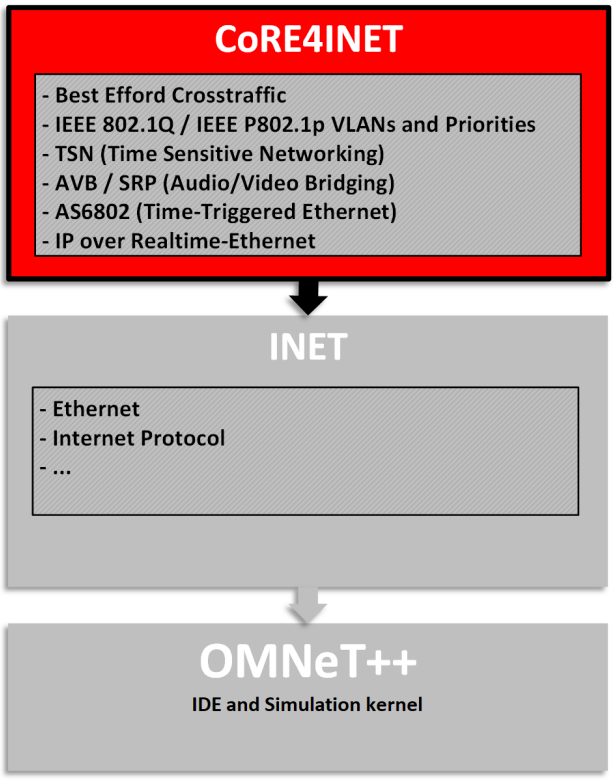

Figure 5.1. Overview of the simulation environment. Source: [7]

– AS6802 TTEthernet (with one time-triggered message and some best-effort messages to allow cross traffic), used in powertrain, chassis domains, autonomous driving, X-by-wire applications to ensure latency of less than 100μs and jitter in the order of microseconds;

– Best-effort frames, allowing cross-traffic by increasing payloads, to analyse the robustness of real-time Ethernet protocols (i.e. AVB, TTEthernet). In general cross-traffic increases latency by 500% and jitter by 14x compared to having no background tasks.[57]

Following the simulation workflow, shown in Figure 5.2, file *.andl* was written:

```
network small_network {
    devices {
      node node1;
      node node2;
      node node3;
      switch switch1;
    }
```

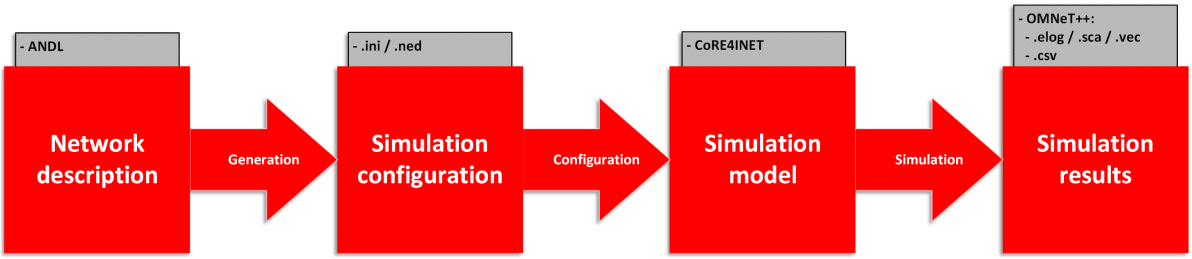

Figure 5.2. Simulation workflow from network description (*.andl* file) to result analysis (*.elog*/*.sca*/*.vec* files). Source: [6]

```
connections {
  segment default {
    node1 <--> {ethernetLink link1 {bandwidth 100Mb/s;
                                    length 20m;}} <--> switch1;
    node2 <--> {ethernetLink link2 {bandwidth 100Mb/s;
                                    length 10m;}} <--> switch1;
    node3 <--> {ethernetLink link3 {bandwidth 100Mb/s;
                                    length 5m;}} <--> switch1;
  }
}

communication {
  message stream1 {
    sender node1;
    receivers node3;
    payload 350B;
    period 125us;
    mapping {
      default : avb {id 1; srClass A;};
    }
  }
  message stream2 {
    sender node2;
    receivers node3;
    payload 350B;
    period 250us;
    mapping {
      default : avb {id 2; srClass B;};
    }
  }
```

```
    message tt_traffic {
      sender node1;
      receivers node2, node3;
      payload 46B;
      period 5ms;
      mapping {
        default : tt { ctID 100; };
      }
    }
    message crosstraffic1 {
      sender node1;
      receivers node2;
      payload 1500B;
      period uniform(200us,500us);
      mapping {
        default : be;
      }
    }
    message crosstraffic2 {
      sender node2;
      receivers node3;
      payload 1500B;
      period uniform(200us,500us);
      mapping {
        default : be;
      }
    }
    message crosstraffic3 {
      sender node3;
      receivers node1;
      payload 1500B;
      period uniform(200us,500us);
      mapping {
        default : be;
      }
    }
  }
}
```

The network, called "*small_network*" is composed by three nodes and one switch, everything connected with links of $100Mb/s$ as bandwidth (100Base-T1) of different lengths ($20m$, $10m$, $5m$), as shown in Figure 5.3. A reasonable propagation delay of maximum $5ns/m$ of latency is considered.

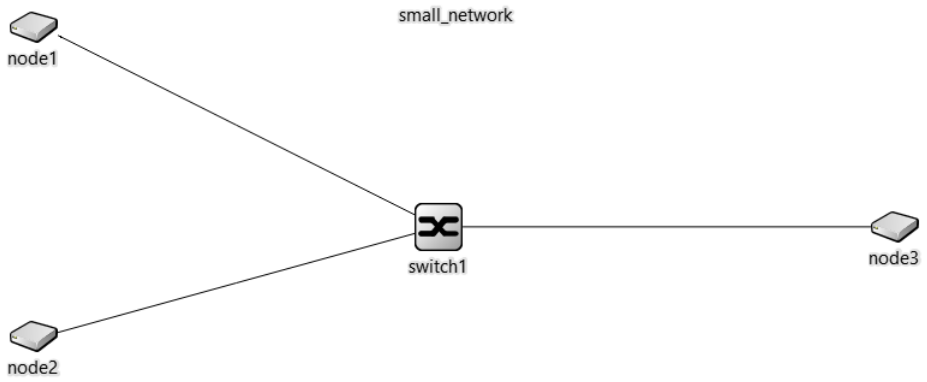

Figure 5.3. "small_network" topology.

Node1 sends to *node3* AVB messages of class A (payload $350Byte$) every $125\mu s$, instead *node2* sends to *node3* AVB messages of Class B (payload $350Byte$) every $250\mu s$. Regarding TTE, node1 sends to both *node2* and *node3* Time-Triggered messages (payload $46Byte$) with $5ms$ as period, instead *node1* sends messages to *node2* in Best-Effort manner with uniform period between $200\mu s$ and $500\mu s$. Same Best-Effort messages are sent from *node2* to *node3* and from *node3* to *node1*.
By combining AVB and TTEthernet traffic, as in Figure 5.4, obviously, it is expected that the *tt_traffic* (Time-Triggered) messages have highest priority overall. Then *stream1* (class A AVB) will have higher priority than *stream2* (class B AVB). Finally *crosstraffic1,2,3* messages will have lowest priority due to their Best-Effort mode, suitable for all legacy IEEE 802.3 Ethernet traffic without any requirement in terms of time (QoS).

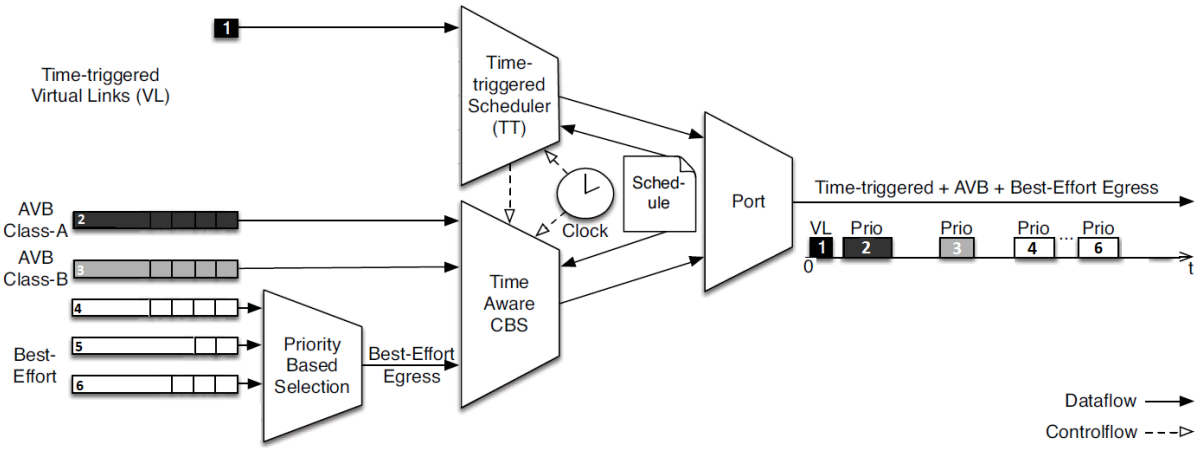

Figure 5.4. AVB, TTEthernet and Best-Effort traffic together. Source: [50]

By generating the network from the *.andl* file, the *.ini* and *.ned* files are created one for each node and switch and one for the entire network. Inside every *.ned* file are imported from libraries (C++) all the pieces to implement the protocols and the stack, instead inside *.ini* file are present all configuration parameters such as the node name, payload, period, MAC, type of message (AVB, TT or BE) and so on.
As an example file *.ned* of a node is shown, graphically, in Figure 5.5, instead file *.ned* of a switch is shown in Figure 5.6. Regarding file *.ini*, below, in Figure 5.7, is shown that of node1 (the others change only in the type of message sent/received and in the time period).

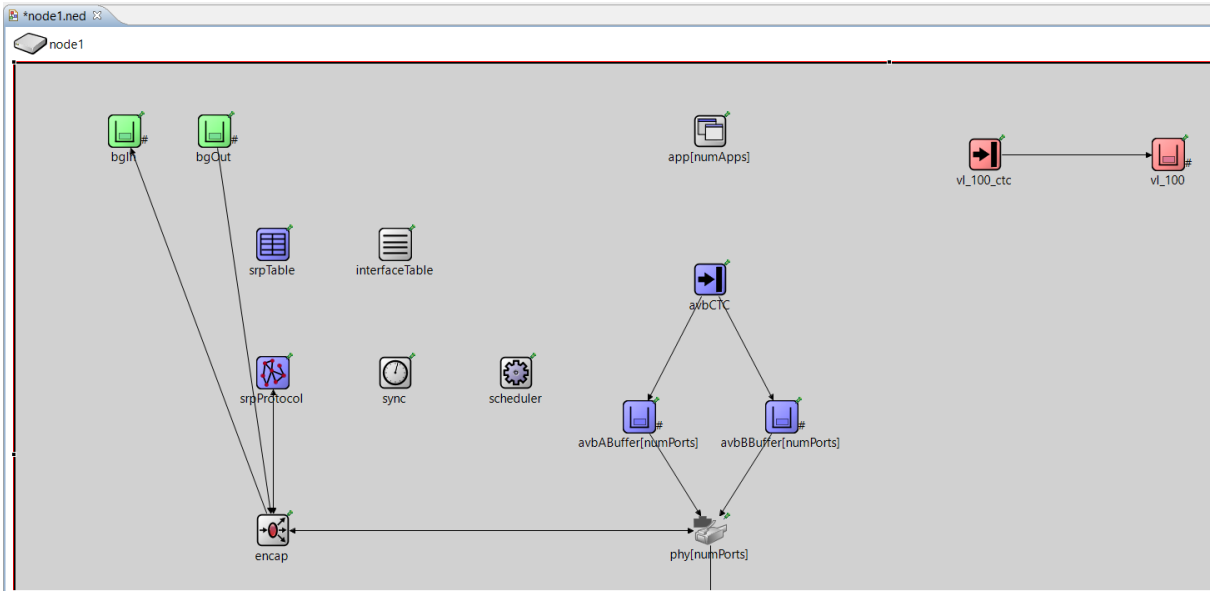

Figure 5.5. *Node1* file *.ned*.

Inside *switch1_AddresTable.txt*, read from *switch.ini*, are contained all MAC addresses, in this case:

C6-92-3E-8F-18-8F node1
5A-0E-48-39-2D-28 node2
BA-51-C1-88-8B-DD node3

The network has been simulated by testing different payloads of the Best-Effort messages to check cross-traffic. Values used are: $64B$, $128B$, $256B$, $512B$, $1024B$, $1280B$, $1500B$.
Figure 5.8 shows the transit of packets starting from 5ms (time at which the first TTE packet is generated) from one node to another passing through the switch.
In Figure 5.9 is shown the latency for AVB and TTE communications for increasing

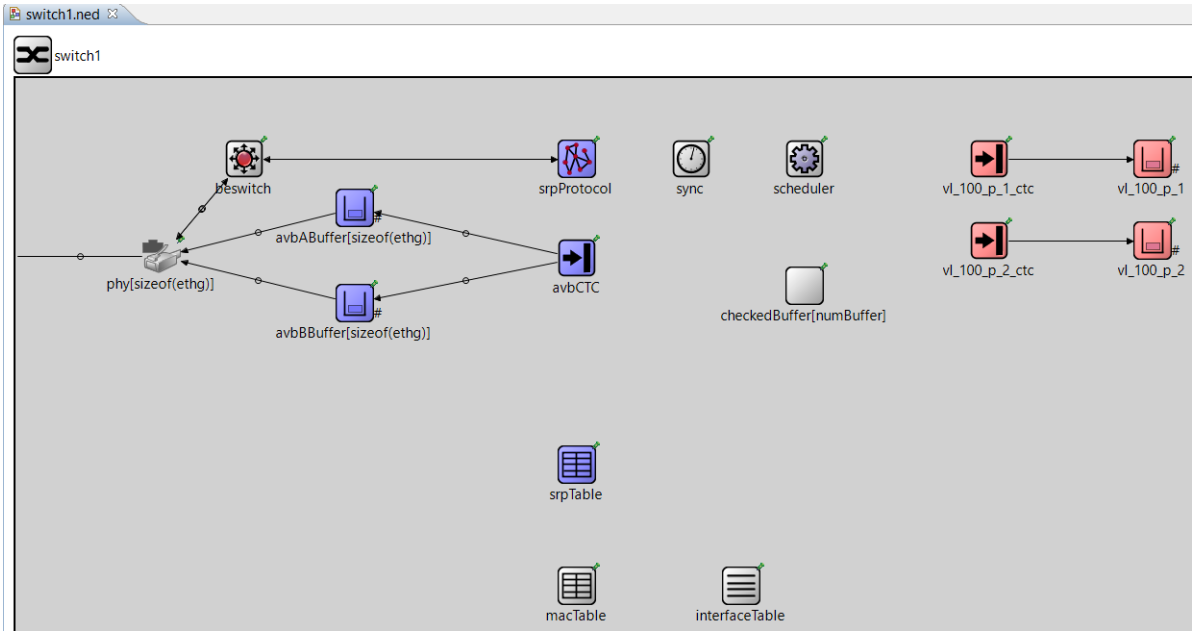

Figure 5.6. Switch file *.ned*.

Cross Traffic frame size of Best-Effort messages. From the results it can be seen that TTE has constant very low latency regardless of the increase in cross-traffic and remains independent from the other streams respecting its determinism. Instead, latency below $2ms$, for AVB suitable applications, is equally guaranteed, but the increase in cross-traffic has a strong impact.
This configuration has a balanced throughput for each node as shown in Figure 5.10 and obviously there is no dropped packet.
Now trying to show the limit of AVB and TTEthernet protocols, some other cross-traffic message and AVB stream is added to overload the whole network. In this case $100Mbps$ bandwidth is nearly saturated as shown in Figure 5.11, in fact there is some AVB packet dropped. Regarding to latency of TTEthernet is still guaranteed $(41\mu s, 42\mu s, 43\mu s)$ also in case of an overloaded network, because its fully deterministic and predictable communication delays and precisions are certainly guaranteed and predefined. The traffic generated by critical messages will always take precedence over traffic generated by non-critical messages.
Instead, regarding AVB, latency grows and many messages are dropped when adding more messages to overload the network.
Summarizing, Table 5.1 shows the results of the simulations mentioned above.

node1.ini

```
network = small_network

**.node1.phy[*].mac.address = "C6-92-3E-8F-18-8F"

**.node1.numApps = 4

# Sink app for message "crosstraffic3":
**.node1.app[3].typename = "BGTrafficSinkApp"
**.node1.app[3].srcAddress = "BA-51-C1-88-8B-DD"

# Source app for AVB message "stream1":
**.node1.app[0].typename = "AVBTrafficSourceApp"
**.node1.app[0].streamID = 1
**.node1.app[0].srClass = "A"
**.node1.app[0].intervalFrames = 1
**.node1.app[0].payload = 350Byte
# Source app for TT message "vl_100":
**.node1.app[1].typename = "TTTrafficSourceApp"
**.node1.app[1].action_time = 4980100ns
**.node1.app[1].payload = 46Byte
**.node1.app[1].ct_id = 100
**.node1.app[1].period = "period[1]"
**.node1.app[1].buffers = "vl_100"

# Source app for BE message "crosstraffic1":
**.node1.app[2].typename = "BGTrafficSourceApp"
**.node1.app[2].destAddress = "5A-0E-48-39-2D-28"
**.node1.app[2].payload = intWithUnit(uniform(1500Byte, 1500Byte))
**.node1.app[2].sendInterval = uniform(2.0E-4s, 5.0E-4s)
**.node1.app[2].startTime = 0 s

**.node1.bgIn.destination_gates = "app[3].in"
**.node1.phy[0].shaper.tt_buffers = "vl_100"

# Virtual link input configuration for TT message "vl_100":
**.node1.vl_100_ctc.receive_window_start = sec_to_tick(0ns)
**.node1.vl_100_ctc.receive_window_end = sec_to_tick(0ns)
**.node1.vl_100_ctc.permanence_pit = sec_to_tick(0ns)

# Virtual link output configuration for TT message "vl_100":
**.node1.vl_100.destination_gates = "phy[0].TTin"
**.node1.vl_100.ct_id = 100
**.node1.vl_100.period = "period[1]"
**.node1.vl_100.sendWindowStart = sec_to_tick(100ns)
**.node1.vl_100.sendWindowEnd = sec_to_tick(7781ns)
```

Figure 5.7. *Node1* file *.ini*.

Time	Relevant Hops	Name	Kind	Length
0.004999965	node1 --> switch1	vl_100	ETH: C6-92-3E-8F-18-8F > 03-04-05-06-00-64	(64 bytes)
0.005006685	node1 --> switch1	Stream 1	ETH: C6-92-3E-8F-18-8F > AB-AA-00-00-00-01	(372 bytes)
0.005015745124	switch1 --> node2	vl_100	ETH: C6-92-3E-8F-18-8F > 03-04-05-06-00-64	(64 bytes)
0.005015745124	switch1 --> node3	vl_100	ETH: C6-92-3E-8F-18-8F > 03-04-05-06-00-64	(64 bytes)
0.005022465124	switch1 --> node2	Best-Effort Traffic	ETH: C6-92-3E-8F-18-8F > 5A-0E-48-39-2D-28	(1518 bytes)
0.005038045	node1 --> switch1	Stream 1	ETH: C6-92-3E-8F-18-8F > AB-AA-00-00-00-01	(372 bytes)
0.005039585422	switch1 --> node3	Stream 2	ETH: 5A-0E-48-39-2D-28 > AB-AA-00-00-00-02	(372 bytes)
0.005070945422	switch1 --> node3	Stream 1	ETH: C6-92-3E-8F-18-8F > AB-AA-00-00-00-01	(372 bytes)
0.005091838836	switch1 --> node1	Best-Effort Traffic	ETH: BA-51-C1-88-8B-DD > C6-92-3E-8F-18-8F	(1518 bytes)
0.005102305422	switch1 --> node3	Best-Effort Traffic	ETH: 5A-0E-48-39-2D-28 > BA-51-C1-88-8B-DD	(1518 bytes)
0.005142593611	node2 --> switch1	Stream 2	ETH: 5A-0E-48-39-2D-28 > AB-AA-00-00-00-02	(372 bytes)
0.005162765	node1 --> switch1	Stream 1	ETH: C6-92-3E-8F-18-8F > AB-AA-00-00-00-01	(372 bytes)
0.005192592844	node3 --> switch1	Best-Effort Traffic	ETH: BA-51-C1-88-8B-DD > C6-92-3E-8F-18-8F	(1518 bytes)
0.005194125	node1 --> switch1	Best-Effort Traffic	ETH: C6-92-3E-8F-18-8F > 5A-0E-48-39-2D-28	(1518 bytes)
0.005225345422	switch1 --> node3	Stream 1	ETH: C6-92-3E-8F-18-8F > AB-AA-00-00-00-01	(372 bytes)
0.005237516294	node2 --> switch1	Best-Effort Traffic	ETH: 5A-0E-48-39-2D-28 > BA-51-C1-88-8B-DD	(1518 bytes)
0.005288706214	switch1 --> node3	Stream 2	ETH: 5A-0E-48-39-2D-28 > AB-AA-00-00-00-02	(372 bytes)
0.005317165	node1 --> switch1	Stream 1	ETH: C6-92-3E-8F-18-8F > AB-AA-00-00-00-01	(372 bytes)
0.005320066214	switch1 --> node3	Stream 1	ETH: C6-92-3E-8F-18-8F > AB-AA-00-00-00-01	(372 bytes)
0.005322697844	switch1 --> node1	Best-Effort Traffic	ETH: BA-51-C1-88-8B-DD > C6-92-3E-8F-18-8F	(1518 bytes)
0.005324305	switch1 --> node2	Best-Effort Traffic	ETH: C6-92-3E-8F-18-8F > 5A-0E-48-39-2D-28	(1518 bytes)
0.005367646294	switch1 --> node3	Best-Effort Traffic	ETH: 5A-0E-48-39-2D-28 > BA-51-C1-88-8B-DD	(1518 bytes)
0.005392590486	node2 --> switch1	Stream 2	ETH: 5A-0E-48-39-2D-28 > AB-AA-00-00-00-02	(372 bytes)
0.005412045	node1 --> switch1	Stream 1	ETH: C6-92-3E-8F-18-8F > AB-AA-00-00-00-01	(372 bytes)
0.005490686294	switch1 --> node3	Stream 1	ETH: C6-92-3E-8F-18-8F > AB-AA-00-00-00-01	(372 bytes)
0.005536685	node1 --> switch1	Stream 1	ETH: C6-92-3E-8F-18-8F > AB-AA-00-00-00-01	(372 bytes)
0.005537726882	switch1 --> node3	Stream 2	ETH: 5A-0E-48-39-2D-28 > AB-AA-00-00-00-02	(372 bytes)
0.005568045	node1 --> switch1	Best-Effort Traffic	ETH: C6-92-3E-8F-18-8F > 5A-0E-48-39-2D-28	(1518 bytes)
0.005569086882	switch1 --> node3	Stream 1	ETH: C6-92-3E-8F-18-8F > AB-AA-00-00-00-01	(372 bytes)
0.005588553089	node3 --> switch1	Best-Effort Traffic	ETH: BA-51-C1-88-8B-DD > C6-92-3E-8F-18-8F	(1518 bytes)
0.005603061171	node2 --> switch1	Best-Effort Traffic	ETH: 5A-0E-48-39-2D-28 > BA-51-C1-88-8B-DD	(1518 bytes)
0.005676066261	switch1 --> node3	Stream 1	ETH: C6-92-3E-8F-18-8F > AB-AA-00-00-00-01	(372 bytes)
0.005691085	node1 --> switch1	Stream 1	ETH: C6-92-3E-8F-18-8F > AB-AA-00-00-00-01	(372 bytes)
0.005698225	switch1 --> node2	Best-Effort Traffic	ETH: C6-92-3E-8F-18-8F > 5A-0E-48-39-2D-28	(1518 bytes)
0.005718658089	switch1 --> node1	Best-Effort Traffic	ETH: BA-51-C1-88-8B-DD > C6-92-3E-8F-18-8F	(1518 bytes)
0.005726101171	node2 --> switch1	Stream 2	ETH: 5A-0E-48-39-2D-28 > AB-AA-00-00-00-02	(372 bytes)
0.005733191171	switch1 --> node3	Best-Effort Traffic	ETH: 5A-0E-48-39-2D-28 > BA-51-C1-88-8B-DD	(1518 bytes)

Figure 5.8. Transit of messages on the network.

Simulations increasing the network overload	***Max AVB Latency*** *(μs)*	***Max TTE Latency*** *(μs)*	***AVB Packets dropped***	***TTE Packets dropped***
1 TTE, 2 AVB	160	41	No	No
1 TTE, 2 AVB, 3 BE	353	41	No	No
1 TTE, 2 AVB, 6 BE	389	41	No	No
1 TTE, 2 AVB, 12 BE	533	42	Yes (15)	No
2 TTE, 4 AVB, 12 BE	693	42	Yes (too high)	No
↓	↓	43	Yes (too high)	No
		43	Yes (too high)	No

Table 5.1. Latencies and packets dropped for various configurations with increasing traffic.

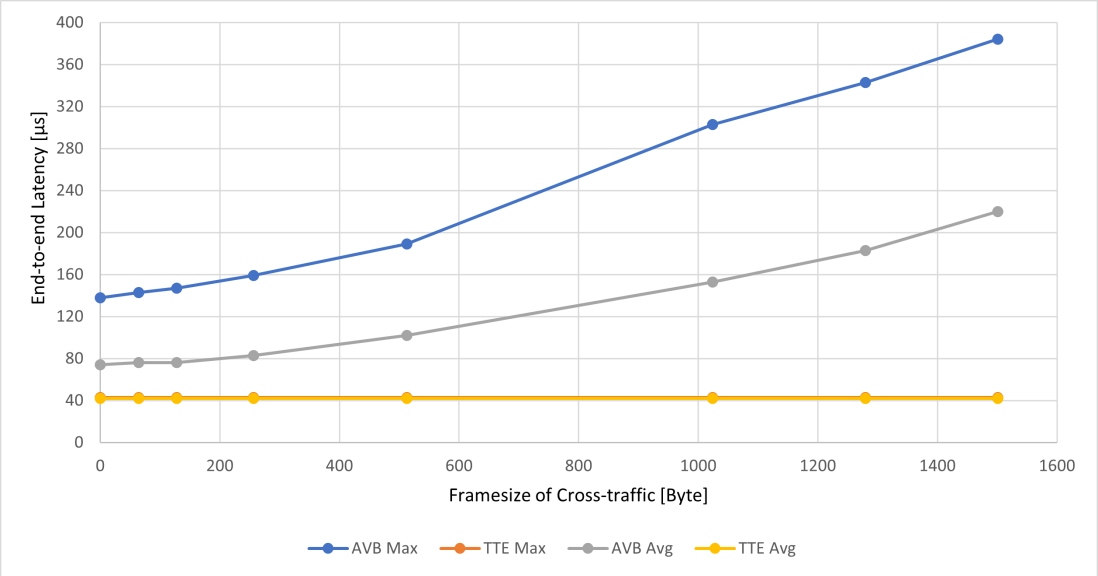

Figure 5.9. Maximum and average end-to-end Latency (AVB and TSN) for different Cross Traffic frame size.

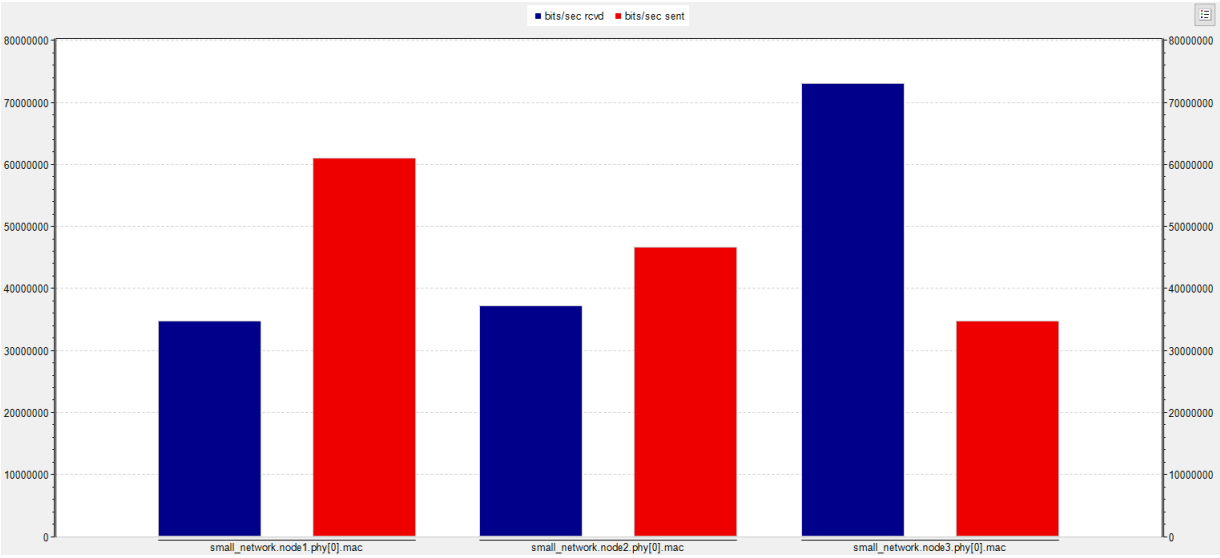

Figure 5.10. Throughput of each node in regular conditions (no overload).

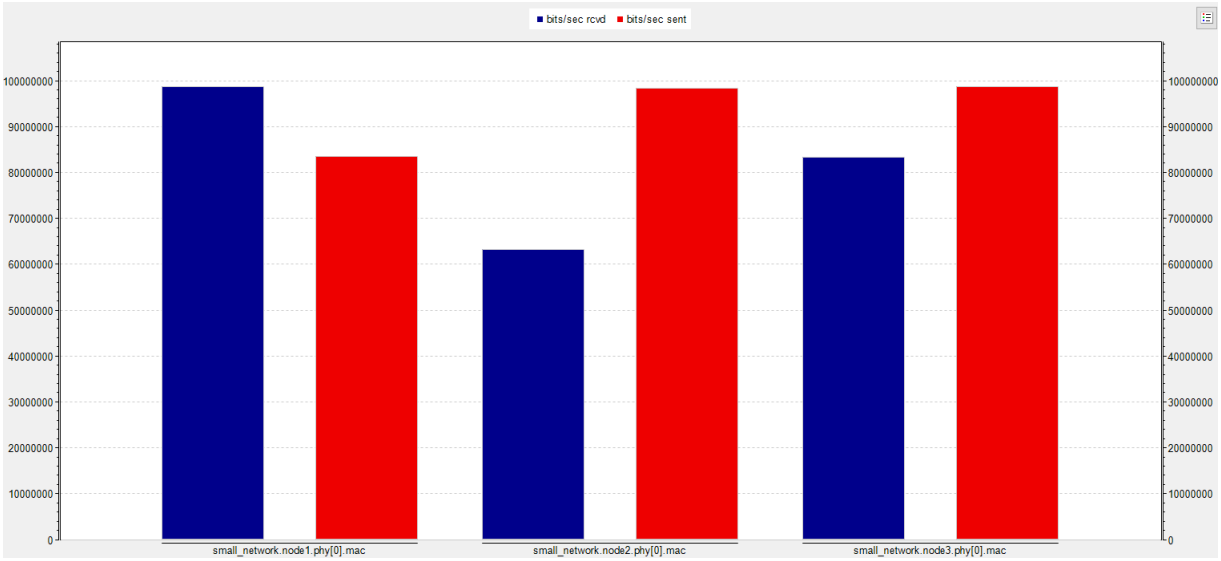

Figure 5.11. Throughput of each node in overload conditions.

Chapter 6

Conclusions and future developments

It is certainly possible to understand that Automotive Ethernet needs to be used in the vehicle as soon as possible due to the above-mentioned benefits. An overview of all protocols, standards and differences with other automotive networks was provided. Testing and validation methodologies have been provided for ECUs interconnected over Ethernet and three nodes exchanging real-time (AVB, TTE) and non-real-time messages (BE) have been simulated, studying their advantages and limits, thanks to open-source frameworks available from a research center at the University of Hamburg.
However, before being put into the vehicle, ECUs on Ethernet need to be tested and physically simulated. For these purposes there are many boards and test benches provided by Vector, Xena, Keysight, NXP, Texas Instruments and other testing companies with a cost ranging from $249 to $6000 for something more professional. A possible future development of this thesis could be to use these boards or test benches to be able to test and simulate, in practice, two or more ECUs connected on Automotive Ethernet with the methodologies mentioned in this thesis. Another important development can be test and simulate mixed network with CAN, FlexRay and Ethernet protocols.

Bibliography

[1] AUTOSAR. Some/ip protocol specification. https://www.autosar.org/fileadmin/user_upload/standards/foundation/1-0/AUTOSAR_PRS_SOMEIPProtocol.pdf.

[2] Bello, Lucia Lo. The case for ethernet in automotive communications. *ACM SIGBED Review*, 8(4):7–15, 2011.

[3] Brunthaler, Sebastian and Waas, Thomas and Kucera, Markus. On verify and validate a next generation automotive communication networka. In *PECCS*, pages 121–127, 2019.

[4] Charles M. Kozierok and Colt Correa and Robert B. Boatright and Jeffrey Quesnelle. *Automotive Ethernet: The Definitive Guide*. Intrepid Control Systems, 2014. 978-0-9905388-06.

[5] Comité International Spécial des Perturbations Radioélectriques. https://webstore.iec.ch/publication/26122&preview, 2016.

[6] CoRE Research Group. Core research group. https://core-researchgroup.de/about/core-group.html.

[7] CoRE Research Group. Core4inet. https://github.com/CoRE-RG/CoRE4INET.

[8] Eclipse. Xtest. https://www.eclipse.org/Xtext/.

[9] Embitel. How some/ip enables service oriented architecture in the new age ecu network. https://www.embitel.com/blog/embedded-blog/how-some-ip-enables-service-oriented-architecture-in-ecu-network.

[10] Embitel Technologies. What is doip protocol and how doip software enables remote vehicle diagnostics? https://www.youtube.com/watch?v=aMDWeKq9gCk.

[11] Ethernet, Time-Triggered. As6802™. *Standard*, 2016.

[12] Fu, Shousai and Chen, HZJ. Time sensitive networking technology overview and performance analysis. *ZTE COMMUNICATIONS*, 16(4), 2018.

[13] Hank, Peter and Suermann, Thomas and Müller, Steffen. Automotive ethernet, a holistic approach for a next generation in-vehicle networking standard. In *Advanced Microsystems for Automotive Applications 2012*, pages 79–89. Springer, 2012.

[14] IEEE 802.3 Ethernet Working Group. 10mb/s single twisted pair ethernet call for interest. https://www.ieee802.org/3/cfi/0716_1/CFI_01_0716.pdf, 2019.

[15] IEEE Working Group. Ieee standard for local and metropolitan area networks - virtual bridged local area networks amendment 12: Forwarding and queuing enhancements for time-sensitive streams. *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pages C1–72, 2010.

[16] IEEE Working Group. Ieee standard for local and metropolitan area networks–virtual bridged local area networks amendment 14: Stream reservation protocol (srp). *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*, pages 1–119, 2010.

[17] IEEE Working Group. Ieee standard for layer 2 transport protocol for time sensitive applications in a bridged local area network. *IEEE Std 1722-2011*, pages 1–65, 2011.

[18] IEEE Working Group. Ieee standard for local and metropolitan area networks - timing and synchronization for time-sensitive applications in bridged local area networks. *IEEE Std 802.1AS-2011*, pages 1–292, 2011.

[19] IEEE Working Group. Ieee standard for a transport protocol for time-sensitive applications in bridged local area networks. *IEEE Std 1722-2016 (Revision of IEEE Std 1722-2011)*, pages 1–233, 2016.

[20] IEEE Working Group. Ieee standard for local and metropolitan area networks – bridges and bridged networks – amendment 26: Frame preemption. *IEEE Std 802.1Qbu-2016 (Amendment to IEEE Std 802.1Q-2014)*, pages 1–52, 2016.

[21] IEEE Working Group. Ieee standard for local and metropolitan area networks – bridges and bridged networks - amendment 25: Enhancements for scheduled traffic. *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)*, pages 1–57, 2016.

[22] IEEE Working Group. Ieee standard for local and metropolitan area networks–bridges and bridged networks – amendment 31: Stream reservation protocol (srp) enhancements and performance improvements. *IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018)*, pages 1–208, 2018.

[23] IEEE Working Group. Ieee standard for local and metropolitan area networks–timing and synchronization for time-sensitive applications. *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pages 1–421, 2020.

[24] IEEE802.3 Working Group. 10g+ automotive ethernet electrical phys. https://www.ieee802.org/3/cfi/0319_1/CFI_01_0319.pdf, March 2019. Call for Interest Consensus Meeting, Vancouver, BC.

[25] International Organization for Standardization. Road vehicles — Diagnostic communication over Internet Protocol (DoIP) — Part 2: Transport protocol

and network layer services. Standard, International Organization for Standardization, December 2019.

[26] Ixia. Aptixia ixautomate rfc benchmarking test suites. https://test4tot.com/wp-content/uploads/2016/05/RFC2889.pdf.

[27] Ixia: a division of Keysight Technologies. https://support.ixiacom.com/resources/automotive-ethernet-overview, 2014.

[28] Jerry Perser and Robert Mandeville. Benchmarking Methodology for LAN Switching Devices. RFC 2889, August 2000.

[29] Jingyue Cao, Siva Thangamuthu. Switched ethernet in time sensitive networking. https://www.win.tue.nl/~johanl/educ/oCPS/oCPS%2004%20TSN.pdf.

[30] JS, Jayadudha and others. Overview of diagnostic over ip (doip), ethernet technology and lightweight tcp/ip for embedded system. *International Journal of Advanced Research in Computer Science*, 4(1), 2013.

[31] Keysight Technologies. Testing automotive ethernet ecu conformance and interoperability. https://www.keysight.com/it/en/assets/7019-0199/solution-briefs/Testing-Automotive-Ethernet-Conformance.pdf.

[32] Keysight Technologies. https://www.keysight.com/it/en/assets/7018-06530/flyers/5992-3742.pdf, 2019.

[33] Kyriakakis, Eleftherios and Lund, Maja and Pezzarossa, Luca and Sparsø, Jens and Schoeberl, Martin. A time-predictable open-source ttethernet end-system. *Journal of Systems Architecture*, 108:101744, 2020.

[34] Lorenzo Passarini. La misura della qos nelle reti a pacchetto ad alta velocità. http://www.dii.unimore.it/~mcasoni/Pres_Klabs.pdf.

[35] Matheus, Kirsten and Königseder, Thomas. *Automotive ethernet*. Cambridge University Press, 2021.

[36] Michael Johas Teener (Mikejt). Avb connections. https://commons.wikimedia.org/wiki/File:AVB-Ethernet-connections.pdf.

[37] Michael Johas Teener (Mikejt). Clock-master. https://upload.wikimedia.org/wikipedia/en/6/64/Clock-Master.pdf.

[38] Michael Johas Teener (Mikejt). Traffic-shaping. https://upload.wikimedia.org/wikipedia/en/d/d3/Traffic-shaping.pdf.

[39] Navet, N. and Song, Y. and Simonot-Lion, F. and Wilwert, C. Trends in automotive communication systems. *Proceedings of the IEEE*, 93(6):1204–1223, 2005.

[40] Navet, Nicolas and Simonot-Lion, Françoise. In-vehicle communication networks-a historical perspective and review. Technical report, University of Luxembourg, 2013.

[41] OPEN ALLIANCE TC8 Members. Open alliance automotive ethernet ecu test specification layer 1. http://opensig.org/download/document/271/OA_Automotive_Ethernet_ECU_TestSpecifications_Version_3.0.zip.

[42] OPEN ALLIANCE TC8 Members. Open alliance automotive ethernet ecu test specification layer 2. http://opensig.org/download/document/271/

OA_Automotive_Ethernet_ECU_TestSpecifications_Version_3.0.zip.

[43] OPEN ALLIANCE TC8 Members. Open alliance automotive ethernet ecu test specification layer 3-7. http://opensig.org/download/document/271/OA_Automotive_Ethernet_ECU_TestSpecifications_Version_3.0.zip.

[44] OPEN ALLIANCE TC8 Members. http://www.opensig.org/download/document/196/9_OA_Automotive_Ethernet_ECU_TestSpecification_v1.pdf, 2016.

[45] OpenSim Ltd. Omnet++. https://omnetpp.org/.

[46] OpenSim Ltd. Omnet++ simulation manual. https://doc.omnetpp.org/omnetpp/SimulationManual.pdf.

[47] OpenSim Ltd. Omnet++ simulation manual. https://inet.omnetpp.org/.

[48] Pattavina, Achille. *Reti di Telecomunicazioni: Networking e Internet*. McGraw-Hill, 2007.

[49] Porter, Donovan. 100base-t1 ethernet: the evolution of automotive networking. *Texas Instruments, Techn. Ber*, 2018.

[50] Soeren Rumpf, Till Steinbach, Franz Korf, and Thomas C Schmidt. Software stacks for mixed-critical applications: Consolidating ieee 802.1 avb and time-triggered ethernet in next-generation automotive electronics. In *2014 IEEE Fourth International Conference on Consumer Electronics Berlin (ICCE-Berlin)*, pages 14–18. IEEE, 2014.

[51] S. Bradner and J. McQuaid. Benchmarking Methodology for Network Interconnect Devices. RFC 2544, March 1999.

[52] Softing Automotive Electronics GmbH. Diagnostic communication over internet protocol. https://automotive.softing.com/fileadmin/sof-files/pdf/de/ae/poster/DoIP_faltblatt_softing.pdf.

[53] Spirent Communications. Automotive ethernet performance testing. https://www.spirent.com/assets/wp_automotive-ethernet-performance-testing.

[54] Spirent Communications. Spirent communications test methodologies layer 2 testing with rfc 2889. https://images10.newegg.com/UploadFilesForNewegg/itemintelligence/WAGAN/Layer_202_20Testing_20with_20RFC_2028891445071217740.pdf.

[55] Spirent Developers, Critical New Test Challenges Facing. Testing automotive ethernet phy. *Engineer*, 2013.

[56] Till Steinbach, Hermand Dieumo Kenfack, Franz Korf, and Thomas C Schmidt. An extension of the omnet++ inet framework for simulating real-time ethernet with high accuracy. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, pages 375–382, 2011.

[57] Till Steinbach, Hyung-Taek Lim, Franz Korf, Thomas C Schmidt, Daniel Herrscher, and Adam Wolisz. Beware of the hidden! how cross-traffic affects quality assurances of competing real-time ethernet standards for in-car communication. In *2015 IEEE 40th Conference on Local Computer Networks*

(LCN), pages 1–9. IEEE, 2015.

[58] Steiner, Wilfried and Bauer, Günther and Hall, Brendan and Paulitsch, Michael and Varadarajan, Srivatsan. Ttethernet dataflow concept. In *2009 Eighth IEEE International Symposium on Network Computing and Applications*, pages 319–322. IEEE, 2009.

[59] Strategy Analytics. `https://www.strategyanalytics.com/`, 2019.

[60] Teledyne LeCroy. `https://teledynelecroy.com/doc/100base-t1-ethernet-appnote`, 2020.

[61] Transition Networks. Service activation test (ethersat) user guide 33540. `https://www.transition.com/wp-content/uploads/2016/06/33540_C-RFC2544-User-Guide.pdf`.

[62] TTTech Computertechnik AG. Time-triggered ethernet - a powerful network solution for multiple purpose. `https://www.tttech.com/wp-content/uploads/TTTech_TTEthernet_Technical-Whitepaper.pdf`.

[63] TÜV NORD Mobilität GmbH and Co. KG. Ethernet ecu conformance testing and network validation. `https://assets.vector.com/cms/content/events/2019/vAES19/vAES19_06_Olschweski_TUEV.pdf`.

[64] Vector Informatik GmbH. Learning module automotive ethernet. `https://elearning.vector.com/mod/page/view.php?id=149`.

[65] Vector Informatik GmbH. Tc8 conformance testing of automotive ethernet networks. `https://assets.vector.com/cms/content/events/2019/vTES19/50_Lectures/10_Slides_Dr_Heiner_HILD_Vector_Testing_Symposium_2019_EN.pdf`.

[66] Vector Informatik GmbH. `https://assets.vector.com/cms/content/know-how/can/Slides/CAN_FD_Introduction_EN.pdf`, 2018.

[67] Vector Informatik GmbH. `https://assets.vector.com/cms/content/know-how/_technical-articles/Pressbook_EN_2018.pdf`, 2018.

[68] Antonio Virdis and Michael Kirsche. Recent advances in network simulation. *EAI/Springer Innovations in Communication and Computing*, 2019.

[69] Wikipedia. Ttethernet. `https://en.wikipedia.org/wiki/TTEthernet`.

[70] Xena Networks. Automotive ethernet. `https://xenanetworks.com/whitepaper/automotive-ethernet/`.

[71] Zhao, Lin and He, Feng and Li, Ershuai and Lu, Jun. Comparison of time sensitive networking (tsn) and ttethernet. In *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pages 1–7. IEEE, 2018.

[72] Zimmerman, G and Jones, P and Lewis, J and Beruto, P and Graber, S and Stewart, H. Ieee p802. 3cg 10mb/s single pair ethernet: A guide. In *Task Force meeting materials*, 2019.

Acknowledgements