Ethernet and IP for Automotive E/E-Architectures

–

Technology Analysis, Migration Concepts and Infrastructure

Der Technischen Fakultät der
Universität Erlangen-Nürnberg
zur Erlangung des Grades

D O K T O R - I N G E N I E U R

vorgelegt von

Andreas Kern

Erlangen 2012

Als Dissertation genehmigt von
der Technischen Fakultät der
Universität Erlangen-Nürnberg

Tag der Einreichung: ..... 5. November 2012
Tag der Promotion: ..... 21. Dezember 2012
Dekanin: ..... Prof. Dr.-Ing. habil. Marion Merklein
Berichterstatter: ..... Prof. Dr.-Ing. Jürgen Teich
..... Prof. Dr.-Ing. Ulrich Heinkel

Acknowledgments

I am sincerely and heartily grateful to my advisor, Prof. Dr.-Ing. Jürgen Teich, for the support, guidance, and encouragement he showed me throughout my dissertation writing. I am sure that it would not have been possible without his scientific and technical advice. Beside this, I would like to thank my friend and advisor from Daimler AG, Dr.-Ing. Thilo Streichert, for his support, advice, and guidance over the past three years. He always found the time for discussing the concept proposals, reviewed my publications, and encouraged me during my academic work with all its ups and downs. I would also like to express my sincere thanks to Dr.-Ing. Vera Lauer and Dr.-Ing. Helmut Leier from Daimler AG for their support during my research work at the Daimler Group Research Center. Moreover, I would like to show my gratitude to all the students who designed, implemented, and evaluated parts of the different concepts during their internships or final papers. Finally, I am truly indebted and thankful to my parents Anton and Rosemarie Kern, to my girlfriend Jessica Schwandt, and to all my friends for their support and encouragement. This work here was in cooperation with Daimler AG and supported in part by the Federal Ministry of Education and Research under project SEIS.

Andreas Kern
Roth, 2012

Contents

1. Introduction **1**

1.1. Contributions 5

1.1.1. Technology Safeguarding 5

1.1.2. Integration and Migration Concepts 7

1.1.3. Simulation and Pre-production Testing 8

1.2. Overview 9

2. Technology Safeguarding of Ethernet/IP-based Automotive Communication Networks **11**

2.1. General Overview about Ethernet, IP, and AVB 13

2.1.1. History of Ethernet 13

2.1.2. The Ethernet Standard 14

2.1.3. Internet Protocol (IP) 16

2.1.4. Transport Protocols UDP, TCP, and IEEE 1722 17

2.1.5. Ethernet AVB 18

2.2. Analysis of Bit and Residual Error Rates 21

2.2.1. Fundamentals 21

2.2.2. Related Work 23

2.2.3. Error Detection Concept 27

2.2.4. Error Type Classification 32

2.2.5. Bulk Current Injection Results 33

2.2.6. Vehicle Integration and Results 36

2.3. Accuracy of Packet-based Time Synchronization Algorithms 41

2.3.1. Fundamentals 41

2.3.2. Related Work 45

2.3.3. Test Environment 47

2.3.4. Test Cases 51
2.3.5. Experimental Results 53
2.4. Throughput Performance Aspects of Resource-limited Systems . . . 58
2.4.1. Fundamentals 58
2.4.2. Related Work 61
2.4.3. Throughput Tuning: System Variants at Transmitting Side . 64
2.4.4. Test Environment 71
2.4.5. Experimental Results 73
2.5. Summary 79

3. Integration and Migration Concepts for Ethernet/IP-based E/E-Architectures 81

3.1. Fundamentals 84
3.1.1. Overview Automotive Communication Technologies 84
3.1.2. Functional Principle of Credit-based Shaping 87
3.1.3. Introduction to Electric and Electronic Architectures 87
3.1.4. Automotive Gateways 88
3.2. Ethernet-based Gateway Strategies 90
3.2.1. Related Work 90
3.2.2. Controller Area Network 91
3.2.2.1. Transformation Concept 91
3.2.2.2. Test Setup 98
3.2.2.3. Experimental Results 98
3.2.3. FlexRay 105
3.2.3.1. Transformation Concept 105
3.2.3.2. Test Setup 117
3.2.3.3. Experimental Results 117
3.3. Replacement Approaches 121
3.3.1. Controller Area Network 121
3.3.1.1. Technology Differences 121
3.3.1.2. Migration Concept 122
3.3.1.3. Experimental Results 130
3.3.2. FlexRay 133
3.3.2.1. Technology Differences 133

3.3.2.2. Migration Concept 133
3.3.2.3. Experimental Results 134
3.4. Summary 139

4. Simulation and Pre-production Testing of Ethernet/IP-based E/E-Architectures 141
4.1. Fundamentals 143
4.1.1. Introduction to Discrete Event Simulation 143
4.1.2. Differences in Physical Shared and Switched Technologies . 144
4.2. Related Work 146
4.3. Simulation-based Evaluation of Switched Networks (Case Study) . . 147
4.3.1. Simulation Concept 147
4.3.2. Multi-Camera Use Case 150
4.3.3. Evaluation Results 153
4.4. Restbus Simulation and Evaluation of Switched Networks 155
4.4.1. Ethernet-Test-Switch Concept 157
4.4.2. Restbus Simulation 161
4.4.3. Experimental Results 163
4.5. Summary 168

5. Conclusions and Future Work 171
5.1. Future Work 175

A. German Part 177

Bibliography 195

Author's Own Publications 209

Acronyms 211

Symbols 215

Index 217

1. Introduction

Todays premium class vehicles implement a variety of distributed applications covering all areas of the vehicle, such as *engine*, *chassis*, *body*, *comfort*, as well as *driver assistance* functions. Most of these systems are interconnected via a common network infrastructure, the so-called *Electric and Electronic architecture* (E/E-architecture) of the vehicle which includes not only the communication but also the power distribution, physical placement of components, and the mapping of functionality on these components. This architecture includes different automotive communication technologies and gateways to enable cross-network communication. Typical communication technologies to interconnect *Electronic Control Units* (ECUs) are *Local Interconnect Network* (LIN) [LIN10], *Controller Area Network* (CAN) [CAN91], *FlexRay* (FR) [Fle10], *Media Oriented Systems Transport* (MOST) [MOS10], and *Low Voltage Differential Signaling* (LVDS) [LVD95]. The general benefits of using a common E/E-architecture are among others the possibility of reusing sensor data for different applications, the optimization of the wiring harness (for example the avoidance of parallel cabling in the same installation spaces), the simple extensibility when adding new functions, and the support for having different expansion stages.

Current predevelopment activities of *Original Equipment Manufacturers* (OEMs) indicate that the number and complexity of such distributed applications will further increase and this will lead to more complex networking requirements for the underlying E/E-architecture with their communication technologies [SKB+11•]. Figure 1.1 highlights the evolution of the E/E-architectures of the Mercedes Benz E-class type series as an example. It is shown that the number of ECUs, busses, and signals has considerably increased in the past. In the future, this trend will continue. In addition, it is assumed that the throughput requirements will also increase drastically due to the fact that more and more complex sensors will arise. Examples are object-driven radar sensors or image-driven cameras.

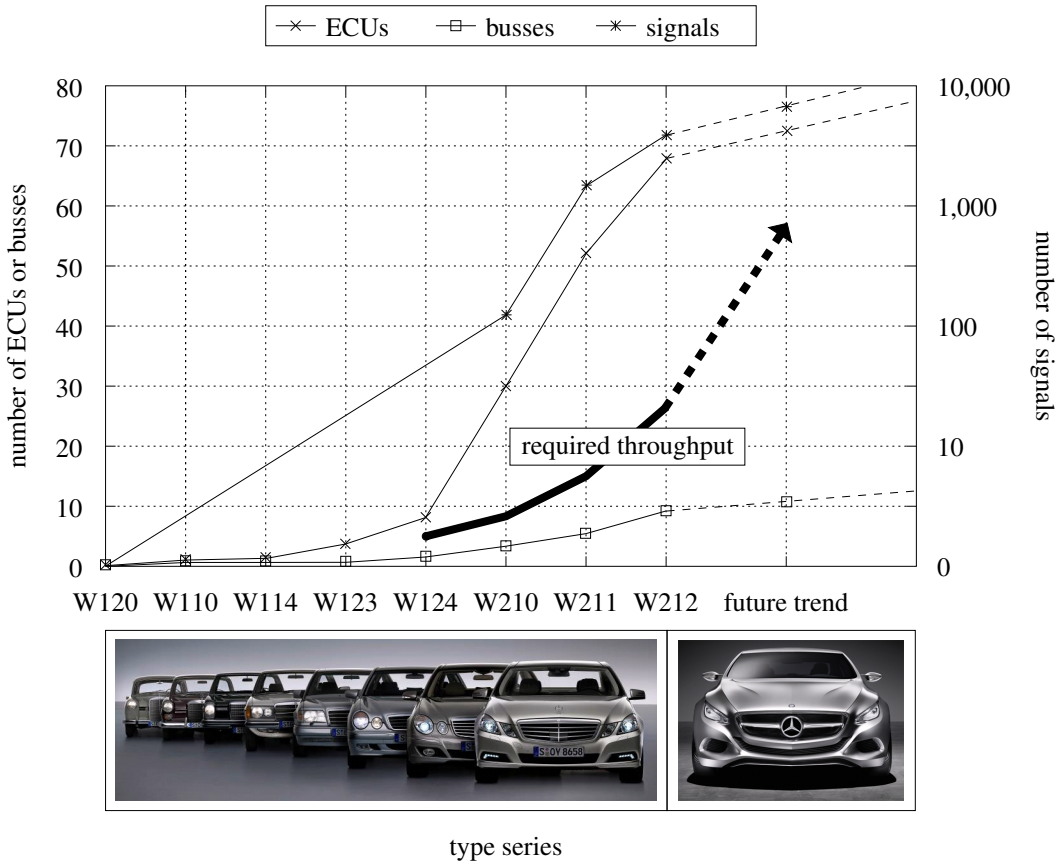


Figure 1.1.: The evolution of the E/E-architectures over all Mercedes Benz E-class type series. The number of mounted ECUs and busses as well as the number of defined signals are presented for each type series, together with the trend for future vehicles. In addition, the changes in throughput requirements are estimated [Rei10].

Ethernet [IEE12b], the *Internet Protocol* (IP) [Req81a], and higher layer protocols like *User Datagram Protocol* (UDP) [Pos80] or *Transport Control Protocol* (TCP) [Req81c] could be one solution to meet the future networking requirements of next generation automotive E/E-architectures. Ethernet is standardized by the *Institute of Electrical and Electronics Engineers* (IEEE) 802.3 working group [IEE12b] and consists of several standards which define the physical and data link layer. The most well-known standards are 10Base-2 (*Thin Ethernet*), 10Base-T (*Ethernet*), 100Base-T (*Fast Ethernet*), and 1000Base-T (*Gigabit Ethernet*). Most of the higher level protocols like IP or UDP are defined by *Request for Comments* (RFCs) [RFC12] and are accepted as world-wide standards.

The main reasons for focusing on Ethernet/IP[1] within the automotive context are that Ethernet/IP is already set for a number of automotive applications communicating between the vehicle and the external world, the discussion of using Ethernet for internal communication within different contexts in the automotive community, and the identification of possible technology transfers from Ethernet consumer and industry standards to automotive standards.

Examples for external communication are *vehicle diagnostics*, *car-2-car communication*, *software download to ECUs*, and *charging of electrical vehicles*. In the field of diagnostics, a new *International Organization for Standardization* (ISO) [OSI12] standard called *Diagnostics over Internet Protocol* (DoIP) [ISO11a] is being developed. It defines the usage of Ethernet and IP to provide a uniform interface and protocols for diagnostic tester. In the car-2-car [Car12] section, the IP protocol is set to guarantee a uniform communication protocol between the car and the external world. External partners can be other cars, traffic lights, or other infrastructure elements. The usage of Ethernet for software download speeds up the service times significantly by removing the todays throughput bottleneck of the connection between the tester and the vehicle. Todays solutions are not standardized and specific to certain OEMs. Powerline [IEE10a] and IP are also in discussion for the charging interface of electrical vehicles to enable payment systems, internet access, and additional applications during the charge process. These topics are summarized under the term *Smart Charging* [ISO11d].

For vehicle internal communication, Ethernet will be standardized as a lead technology for camera links and is discussed for alternative E/E-architecture topologies. The potential benefits when using Ethernet for connecting cameras are cost advantages when using unshielded cabling and the availability of switches to build more complex topologies instead of just having discrete connections between two devices as in the case of LVDS for example. Beside the infotainment domain, the limited physical throughput of today's shared communication technologies of at most 10 Mbit/s could also be an enabler for the introduction of Ethernet by replacing current technologies for intra-domain communication. In addition, Ethernet is evaluated for building alternative E/E-architecture topologies. An example is the backbone

[1] In the following, the term "Ethernet/IP" denotes the Ethernet standards specified in IEEE 802.3 including additional layer one and two services like IEEE 802.1 as well as all higher level protocols like IP or the transport protocols UDP, TCP, or IEEE 1722.

architecture [BRR11], where Ethernet is used as a high-speed data connection to interconnect different car domains.

Another important point is the evaluation of the adaptability of existing Ethernet technologies from different industrial sectors. The adaption of an existing technology offers cost benefits, a reduced time to market, and a faster development compared to the development of a complete new automotive communication technology. Currently, Ethernet is used in the consumer, industrial automation, telecommunication, and avionic sectors. In the industrial section, for example, more than a dozen different Ethernet-based technologies [Tim12, Eth12a, Pro12, Eth12c, Ser12] have been developed to provide specific requirements like hard real-time or special environmental conditions.

From a technical perspective, Ethernet/IP is also very attractive. Ethernet offers different throughputs from 10 Mbit/s up to 40 Gbit/s for non-automotive applications. Moreover, it provides most different packet sizes ranging from 64 bytes up to 1500 bytes. It has switches for efficient resource utilizations and supports a variety of extensions like virtual networks or priorities to implement *Quality of Service* (QoS). In addition, there exist extensions like Ethernet *Audio Video Bridging* (AVB) [IEE12a] which provide throughput reservation, time synchronization, and special packet shaping mechanisms for transferring time-sensitive data. IP and higher protocols like the transport protocols UDP, TCP or IEEE 1722 [IEE11a] offer a variety of high-level services. For example, TCP enables reliable data connections and IEEE 1722 provides a protocol for streaming data which could be used for camera links. At the application layer, a variety of *middlewares* exist which support several services like *Remote Procedure Calls* (RPCs) [Req76] or *Service Discovery* (SD) [CAG05, CK11b, CK11a]. RPCs are used to enable inter-process communication. With this technique, it is possible to call remote software functions (usually located at other ECUs) within different address spaces without coding the details for this remote call explicitly. Service Discovery is used to announce distributed services within a network. Software components can then dynamically subscribe or unsubscribe for different services to use the functionalities provided by them like remote functions or notifications.

A state of the art summary of the research and development activities on Ethernet/IP for the automotive use case was presented at [Mat11]. In [Fri11, JO11, SBLS11, Kri11, Got11], different OEMs presented their actual research activities.

Future topics are the usage of Ethernet/IP for discrete networking applications like cameras and the research on Ethernet/IP as a high-speed backbone within the network architecture for example. In addition, several tier 1 and tier 2 automotive suppliers presented their research topics and prototypes based on more detailed application scenarios like real-time video or audio distribution within the vehicle [Sin11, Nöb11, Kre11, Pow11, Hoe11]. More general topics like EMC-robustness and a proposal for an automotive middleware solution were presented in [KKM+11, Völ11].

In summary, it is shown that Ethernet/IP is an upcoming and serious network solution for the automotive section. There exists a variety of automotive use cases and the number of researchers as well as people in the industrial sector working on this topic are increasing. Up to now, there are still many open questions concerning the use of Ethernet/IP in the automotive context.

1.1. Contributions

The contributions of this thesis are grouped into the three fields 1) *Technology Safeguarding of Ethernet/IP-based Automotive Communication Networks*, 2) *Integration and Migration Concepts for Ethernet/IP-based E/E-Architectures*, and 3) *Simulation and Pre-production Testing of Ethernet/IP-based E/E-Architectures*. An overview about these topics is given in the following.

1.1.1. Technology Safeguarding of Ethernet/IP-based Automotive Communication Networks

In order to employ a new communication technology within the automotive environment, a variety of basic requirements have to be fulfilled. This thesis focuses on error rate analysis, time synchronization accuracy, and throughput acceleration of IP-based automotive communication links.

Analysis of Bit and Residual Error Rates Residual errors regarding communication networks are packets which pass the error check mechanism, though including transmission errors. The result is that applications will process faulty data and this could influence the behavior of a given application. Methods for determining the residual error rates are *direct code analysis*, *transformed code analysis*, *stochastic*

automata, and *monte carlo simulation*. Unfortunately, the analytical approaches will only work for small packet sizes efficiently and the quality of the results obtained from the simulation-based approach depends on the number of simulated samples compared to the number of undetectable error patterns. An important input for all calculations is the knowledge of the frequency and distribution of bit errors. Without this input value, all analytical approaches fail. Therefore, this thesis presents a concept which is designed and implemented to determine the occurrence of bit errors for a given Ethernet link and the residual error rates are determined depending on the error check mechanisms from Ethernet, IP, and UDP. Vehicle measurements for critical installation paths show isolated bit errors and their distributions as well as the absence of residual errors during the whole measurements.

Accuracy of Packet-based Time Synchronization Algorithms Todays automotive communication technologies are mainly based on a physical shared media. In such cases, the synchronization mechanism is based on the simultaneous reception of a common bit pattern or frame at all nodes. When having full-duplex switched Ethernet networks, packet-based synchronization algorithms take place to establish a global time base. Typical algorithms are IEEE 1588-2002 [IEE02], IEEE 1588-2008 [IEE08a], and IEEE 802.1AS [IEE10b]. The accuracy of such algorithms also depends on the robustness of the used oscillators. Up to now, these algorithms are applied in environments like data centers or PCs where the temperature conditions are very stable. In order to show the applicability in harsh automotive environments, this thesis evaluates the synchronization accuracy of IEEE 802.1AS networks under varying temperature conditions. An IEEE 802.1AS capable network is set up to measure the synchronization accuracy under different climate conditions. It is shown that packet-based synchronization algorithms keep up with the special climate conditions even when using shock chambers to realize rapid temperature changes. The outcome of the climate chamber measurements are presented in [KZS+11•].

Throughput Performance Aspects of Resource-limited Systems
Measurements presented in [Ker08•] showed that in resource-limited embedded systems, the measured throughput is far from the theoretically possible throughput of the underlying physical layer. Even when using special mechanisms like *Direct Memory Access* (DMA) for copying data inside the device efficiently, the achievable through-

put is still restricted. The main reason is the resource intensive execution of the software network stack for processing the upper layer protocols. In the following, different concepts will be proposed to improve the throughput of an Ethernet/IP link. The different concepts move from software-dependent solutions to hardware accelerated solutions. It is shown, that the implementation of these concepts achieves up to line-rate by disabling functionalities and using additional hardware accelerations. The different concepts to optimize the Ethernet throughput are presented in [KSS+10•].

1.1.2. Integration and Migration Concepts for Ethernet/IP-based E/E-Architectures

When deploying a new communication technology like Ethernet/IP, gateway strategies are essential to enable cross-network communication. In the long run, even replacement approaches which replace current communication technologies by Ethernet/IP are imaginable. The contributions to interconnect or replace the communication technologies CAN and FlexRay are presented in the following.

Gateway Strategies Todays E/E-architecture topologies are typically based on a central gateway which is used to interconnect most of the assembled communication technologies. Therefore, a gateway is equipped with multiple mixed technology interfaces to allow data exchange between the connected subnetworks. By adding Ethernet/IP as another interface to the gateway, concepts to route traffic between todays communication technologies will be necessary. This thesis describes concepts to forward data traffic between CAN and Ethernet/IP as well as FlexRay and Ethernet AVB. Different packaging mechanisms from simple one-to-one mappings to more complex many-to-one mappings are designed and validated by prototypical implementations which are integrated into a current E-class vehicle. In the case of FlexRay, Ethernet AVB mechanisms are used to guarantee defined throughputs between endpoints and to ensure timing constraints. In this case, the FlexRay frames are encapsulated in IEEE 1722 packets. An automotive IEEE 1722 subtype proposal is proposed in [KSG+11•] and the results of the gateway evaluations are presented in [KRST11•].

Replacement Approaches Beside the interconnection of existing communication technologies with Ethernet/IP, the partial or even full replacement of current technologies could be attractive in the long term. A typical use case is the replacement of CAN when having an Ethernet connection in parallel for example. This thesis presents two concepts for replacing CAN and FlexRay by Ethernet/IP. The main differences which have to be observed are the addressing mechanisms, the send behavior, and the payload size. In the case of CAN, a reference one-to-one mapping approach will be compared with an *Evolutionary Algorithm*-based (EA) and a rule-based approach which try to optimize the embedding of several frames into an Ethernet-based packet. All approaches are automated and applied to real network configurations. When replacing FlexRay, *Virtual Local Area Networks* (VLANs) [IEE11b] are used to resolve the addressing problem and the packaging approaches distinguish between in-cycle and multiple-cycle communication. The multiple-cycle approach is typically used for transport protocols over FlexRay. For both scenarios, it is shown, that the replacement of CAN and FlexRay by Ethernet/IP is possible and detailed comparisons of the average frames per packet, the payload-ratio compared to the whole packet, and the necessary Ethernet throughputs are presented for real car network configurations. The results are published in [KST11•].

1.1.3. Simulation and Pre-production Testing of Ethernet/IP-based E/E-Architectures

This part covers the simulation, restbus simulation, and monitoring of point-to-point Ethernet networks as well as the execution of stress test scenarios. Todays automotive evaluation tools are designed to work with a physically shared media. In such cases, the tracing and generation of frames is simply done by adding a test node to the network. When having point-to-point networks and switches, new approaches are necessary for simulation, traffic monitoring, and traffic generation due to not having the complete communication on one single wire. This thesis presents an Ethernet-Test-Switch concept which allows the testing of switched Ethernet networks in automotive environments. In addition, an interface is designed to combine simulation environments and the switch to allow restbus simulation of systems which are partly available in hardware.

Simulation-based Evaluation of Switched Networks Simulation is a common method for evaluating systems in the early stage of development. It is possible to get an understanding of protocol or timing behavior of a network system, even when no target hardware is available. This thesis introduces a concept to simulate Ethernet/IP-based systems including software tasks, the network stack, and different process schedulers with the use of a *Discrete Event Simulator* (DES). All necessary extensions for simulating Ethernet/IP are integrated and validated by different use cases. In summary, it is shown that discrete event simulation is a powerful method to estimate the functional and timing behavior of an Ethernet-based automotive system. The evaluation results of a multi-camera application case study are presented in [RKH+10•].

Restbus Simulation and Evaluation of Switched Networks During the test phase when the first target hardware is available, testing of these systems becomes an initial part of the development process to show the reliability of the system. It is distinguished between systems which are completely available and systems which require restbus simulation when not all the devices of a given system are available. In this area, a concept for an Ethernet-Test-Switch is designed and implemented. It allows the creation of additional traffic to perform stress test, the tracing of the whole network traffic through the switch, and the realization of restbus simulation to test mixed systems consisting of real and simulated devices. The concept, implementation, and evaluation of the introduced Ethernet-Test-Switch is presented in [KZST11•]. In addition, the concept of the Ethernet-Test-Switch is patented in [KSZ11•].

1.2. Overview

This thesis is structured as follows: Chapter 2 deals with technology safeguarding. In this context, different concepts are created to determine the bit and residual error rates of given Ethernet implementations, to measure the accuracy of packet-based synchronization algorithms under different climate conditions, and to accelerate the sending performance of an Ethernet/IP interface. Multiple communication architecture elements are proposed in Chapter 3. CAN/Ethernet and FlexRay/Ethernet AVB gateways are designed, implemented, and evaluated. In addition, replacement strate-

gies are described to replace CAN or FlexRay by Ethernet/IP in the long run. Chapter 4 enters into the field of evaluation. For the early stages of development, a simulation environment is designed which can simulate Ethernet/IP networks. During the test phase of an application, the developed Ethernet-Test-Switch can be used to investigate the behavior of a given implementation. Conclusions and proposals for future work are discussed in Chapter 5.

2. Technology Safeguarding of Ethernet/IP-based Automotive Communication Networks

To introduce a new communication technology such as Ethernet/IP for automotive applications, a defined process has to be passed to evaluate its risks and rewards. One initial part of this process, for example, is to provide the evidence that the work packages listed in Table 2.1 can be fulfilled. Since Ethernet/IP has already been introduced in many different industrial sectors, some work packages can be easily adapted. Other work packages define special automotive-related requirements which have to be checked explicitly. The listed work packages as well as their particularities regarding Ethernet/IP are described in the following.

The *Electromagnetic Compatibility* (EMC) deals with the unwanted effects of electromagnetic energy. It is the goal to verify the error-free operation of different equipment within the same electromagnetic environment. In the case of Ethernet, transceivers, connectors, and wires have to be evaluated to guarantee a meaningful residual error rate which is an important input for upper layer error detection and correction algorithms. Packaging aspects addresses the problem of designing Ethernet-networked ECUs in small packages and installation spaces as well as the optimization of heat dissipation and the mounting of the connector. Examples are small packaged cameras or radar sensors housed in cubes with side lengths of about 0.98 inches (2.5 centimeters). Another important topic is the timing behavior of a communication technology. For example, when using FlexRay with its *Time Division Multiple Access* (TDMA) method for accessing a channel, the latency of a frame can easily be determined. In the case of Ethernet, when having multiple queues per port and switches to enable larger networks, the question of how long it takes to forward a

message through the network cannot be answered in such an easy way. Therefore, a detailed evaluation of the different mechanisms and their combinations has to be done. In general, the power consumption and weight of devices have to be optimized to fit automotive requirements. It is examined whether devices support *partial networking* or other approaches like *pretended networking* [SKSB11]. Partial networking allows the activation and deactivation of parts of the network to save energy. The wake up and shut down is typically controlled by a central device – the central gateway for example. Pretended networking advances partial networking by adding more intelligence into each end node to allow simple interactions without waking up the complete end node. In the case of Ethernet, special Ethernet-related wake up mechanisms have to be evaluated to enable an Ethernet-networked ECU to be part of a partial or pretended network. The computational resource work package deals with the problem of having resource-limited processors and memories in embedded systems [LWH11, LKVZ11]. In contrast, Ethernet has high data rates, large packet sizes, complex error detection mechanisms, and a multitude of high level protocols which need a lot of computing power. The last six work packages are related to economical aspects like cost advantages when using Ethernet/IP, to the availability of parts and

Table 2.1.: The table lists some of the necessary work packages which have to be fulfilled to enable Ethernet/IP as an automotive-qualified communication technology. The work packages affected by this thesis are highlighted [SKB+10•].

No.	General and Application-specific Work Packages
G.1	Electromagnetic Compatibility (EMC)
G.2	Packaging
G.3	Timing Behavior
G.4	Power Consumption and Weight
G.5	Computational Resources
G.6	Economic Efficiency
G.7	Availability of Automotive-qualified Ethernet-Parts, Second-Sourcing
G.8	Availability of Experienced Suppliers
G.9	Partnerships with other OEMs
A.10	Compression Behavior and Pattern Recognition
A.11	Security Aspects

suppliers, and to application-specific topics like the compression together with pattern recognition [Rah09] or security aspects when connecting external devices or devices which are installed outside the body of the vehicle [KKS+10, BGH+10, HK11].

The following section presents an overview of the Ethernet, IP, and AVB protocols (Section 2.1) and the research activities of this thesis addressing the work packages G.1 (Section 2.2), G.3 (Section 2.3), and G.5 (Section 2.4). In the area of physical layer robustness, a concept to measure bit and residual error rates is designed, implemented, and finally evaluated with *Bulk Current Injection* (BCI) and vehicle measurements. An IEEE 802.1AS capable network is set up to measure the synchronization accuracy of distributed clocks under different climate conditions. Finally, a concept for optimizing the transmission throughput of a resource-limited embedded device is presented and evaluated.

2.1. General Overview about Ethernet, IP, and AVB

The following sections present a general overview about Ethernet, IP, higher transport protocols, and additional methods and protocols called Ethernet AVB [IEE12a]. Ethernet AVB enables the transport of time-sensitive data by providing limited worst case latencies for packets and guaranteed throughputs on dedicated links. It uses standard IEEE packet-based synchronization mechanisms [IEE10b] and is based on the standard Ethernet physical layer.

2.1.1. History of Ethernet

On May 22, 1973, Robert M. Metcalfe did the first handwritten annotation of the so-called “Ether” which described the communication across cables, telephone, and radio based on the previously developed *ALOHAnet protocol*. The ALOHAnet, also known as ALOHA, is a computer networking system developed by the university of Hawaii with the idea of using low-cost amateur radio-like systems to interconnect the far-flung campuses of the university [Eth12b]. Figure 2.1 shows the first handwritten drawing of the “The Ether” from Robert M. Metcalfe. In 1977, Metcalfe et al. patented Ethernet under the name “Multipoint Data Communication System with

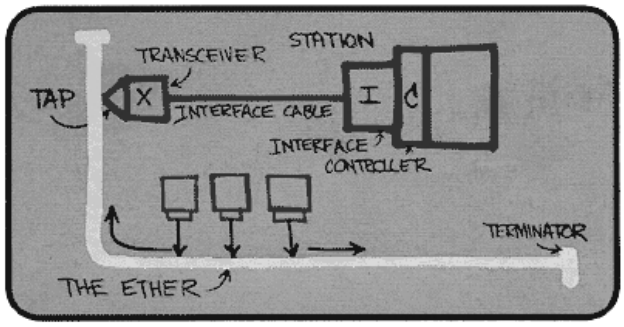


Figure 2.1.: The first handwritten drawing of "The Ether" from Robert M. Metcalfe created in May 1973 [Met94].

Collision Detection". In the following decades, a multitude of Ethernet standards [Rec08] were developed enabling throughputs ranging from 10 Mbit/s up to 40 Gbit/s for present Ethernet implementations. The first standard 10Base-2 called *Thin Ethernet* was released in 1985 using a coaxial cable, and enabled a maximum data rate of 10 Mbit/s. In 1995, *Fast Ethernet* was standardized under the name 100Base-T and allowed a maximum data rate of 100 Mbit/s. It uses *Unshielded Twisted Pair* (UTP) cables for interconnecting the devices. In addition, hubs and switches are introduced to enable larger network topologies. The *Gigabit Ethernet* standard was released in 1999. Current Ethernet standards support up to 40 Gbit/s by using fiber or coaxial cables.

2.1.2. The Ethernet Standard

As many other standards, Ethernet is based on the *Open Systems Interconnection* (OSI) model [Zim80] which groups similar communication functions together and provides defined interfaces between these groups. Figure 2.2 shows the seven layers of the OSI model together with the Ethernet layers and their assignments. Ethernet takes place in the two lowest OSI layers named *physical layer* and *data link layer*. The physical layer consists of the *Medium Dependent Interface* (MDI), the *Physical Medium Attachment* (PMA), the *Attachment Unit Interface* (AUI), and the *Physical Layer Signaling* (PLS). All four parts ensure, that the digital data provided by the data link layer can be processed to enable the transmission of the data via the medium. The definition of the interface between the physical layer and the data link layer is called *Media Independent Interface* (MII). It enables the compatibility to different physical

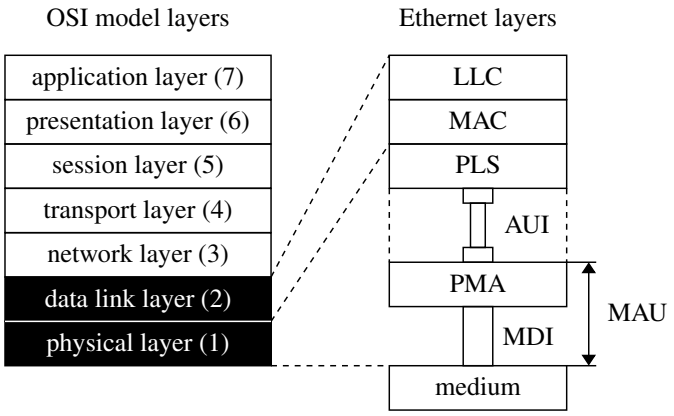


Figure 2.2.: The left side shows the seven OSI model layers. On the right side, the Ethernet layers are presented together with their assignments to the OSI layer model [Rec08].

layer technologies independent from the data link layer. The data link layer consists of the *Media Access Control* (MAC) and *Logical Link Control* (LLC).

It manages the network access by providing arbitration mechanisms and buffering as well as a defined frame format for Ethernet packets. Figure 2.3 shows the frame formats of an untagged standard Ethernet II and a tagged Ethernet packet. Both packets start with a *destination* and *source MAC address* of the transmitter and receiver of the packet. A MAC address is a worldwide unique 6 byte number which clearly identifies every *Network Interface Card* (NIC). The standard address format consists of six groups of two hexadecimal digits, separated by colons. An example is the broadcast MAC address $FF{:}FF{:}FF{:}FF{:}FF{:}FF$. The third common field of all Ethernet frame formats is the 4 byte *Cyclic Redundancy Check* (CRC) field in the end of the packet. It contains an IEEE 802.3 CRC-32 to detect transmission errors. Within the untagged Ethernet II packet, the *type* field specifies the upper layer protocol used by the network layer. The tagged Ethernet packet uses a tag protocol identifier to specify the upper layer protocol plus two additional bytes containing QoS parameters and the virtual network identifier. The upper layer interface to the network layer is typically implemented by a backbone or on-chip bus. Vendor-specific software drivers manage the interaction between the *Operating System* (OS) and the Ethernet controller by reading and writing hardware registers. All remaining layers from three up to seven are independent of the communication technology. The network layer accommodates the popular IP which is described in the following.

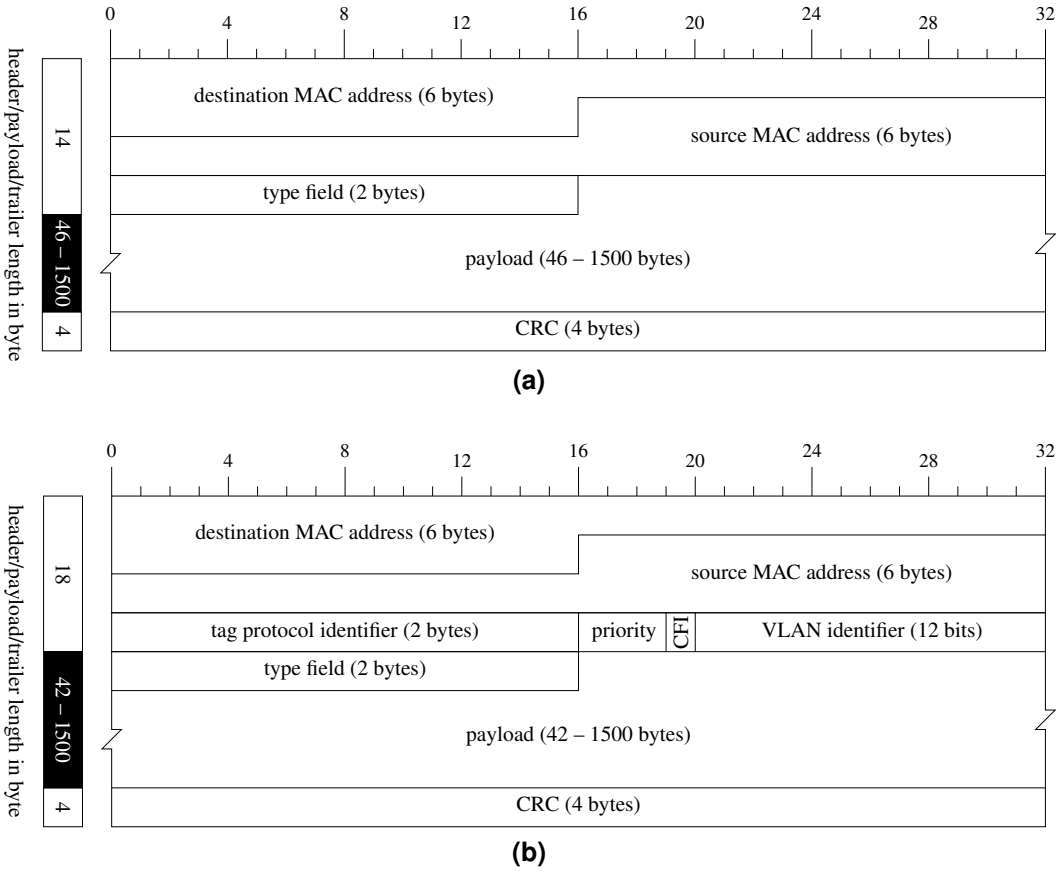


Figure 2.3.: Two Ethernet frame formats: (a) Shows an untagged Ethernet II frame as specified in [IEE08b]. (b) Presents the tagged Ethernet frame format as defined in [IEE06].

2.1.3. Internet Protocol (IP)

The Internet Protocol is the first hardware-independent layer in the OSI network model. In protocol version 4, it contains the logical IP addresses of a transmitter and receiver node, for example, two ECUs within a network. Figure 2.4 shows the complete IP frame format. The *version* field contains the protocol version (in this case 4) and the header length including the optional header fields which is stored in multiples of 32 in the *IP Header Length* (IHL) field. With the *Type Of Service* (TOS) identifier, priorities can be assigned to a packet to influence the routing and forwarding within a router or a switch. The *total length* field represents the length of the entire packet including header and payload with a maximum size of 65535 bytes. *Identification*, *flags*, and *fragment offset* are used to allow IP fragmentation. This mechanism is used

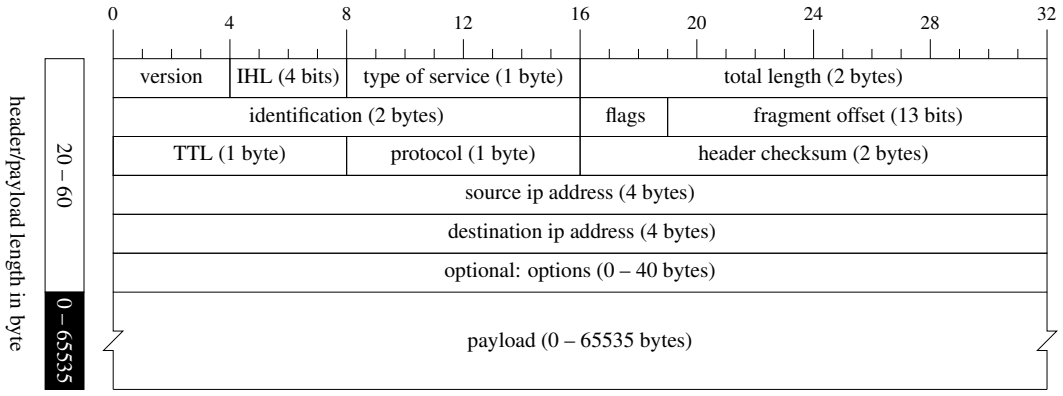


Figure 2.4.: The IP version 4 frame format as defined in [Req81a].

to divide large IP packets into several smaller packets, if endpoints or routers are not able to process large IP packets. Today, however, IP fragmentation is no longer supported by most of the routers. Instead, routers will discard such packets and send a control message back to the source ECU to inform the transmitter of the packets that the chosen packet size is not supported [Req81b]. The *Time To Live* (TTL) counter represents the lifetime of a packet. Every router decrements the value by one. If the value is zero, the packet will be discarded. This prevents from circulating packets. The *protocol* field identifies the upper layer transport protocol, like UDP or TCP and the *header checksum* field is used to detect transmission errors within the header. The last two required header fields are the *source* and *destination IP addresses* of the two communication endpoints. Additionally, it is possible to store additional information within the *options* field. In summary, IP allows logical addressing of endpoints within a large network. Application-dependent addressing is introduced by so-called transport protocols and is discussed in the following.

2.1.4. Transport Protocols UDP, TCP, and IEEE 1722

Transport protocols are located at the transport layer 4 in the OSI model and are the first application-dependent protocols. The major task is the addressing of applications on a network endpoint. In addition, there exist several protocols with most different features beginning from simple protocols like UDP, which only supports application addressing, to more complex protocols like TCP, which is a state-oriented and reliable communication protocol. IEEE 1722 is mainly used for media traffic.

User Datagram Protocol (UDP) This is the most popular protocol for stateless communication. It allows the addressing of application endpoints and supports a header and payload checksum. The complete header is shown in Figure 2.5(a) and has a size of 8 bytes. The application addressing is done with the two attributes *source* and *destination port*. It does not support any reliable transmission of frames. This means for example, that packet loss during transmission is not detected by the protocol and no retransmission is initiated.

Transport Control Protocol (TCP) Reliable full-duplex transmission of packets between two application endpoints is introduced by TCP, the most popular state-oriented transport protocol. It supports upper layer application addressing as well as state-oriented communication with retransmission of packets and ordering of packets to enable reliable communication. The congestion control avoids network overload. Figure 2.5(b) highlights the protocol header with a size of at least 20 bytes. The *port* fields are the same as in the case of UDP. The additional fields are used to exchange the necessary information between the transmitter and receiver to allow reliable communication. With TCP, it is necessary to establish a connection between the two endpoints, before the communication can start. The implementation of the TCP protocol stack is also more complex due to the state orientation, the underlying timing requirements (e.g., retransmission or connection timeouts), and the need for additional buffers to store incoming packets.

IEEE 1722 The IEEE 1722 transport protocol is mainly used for media traffic. Figure 2.5(c) shows the protocol header with a size of 24 bytes. In this case, the application addressing is done by a *stream identifier*. The stream identifier belongs to a media stream which is send by a transmitter to one or more receivers. The *avtp timestamp* contains the presentation time of media samples within the payload section. Today, IEEE 1722 is mainly used to carry audio and video traffic within an AVB network.

2.1.5. Ethernet AVB

Ethernet AVB is being standardized by the IEEE 802.1 task group [IEE12a] in order to support time-sensitive applications in asynchronous Ethernet networks. Ethernet

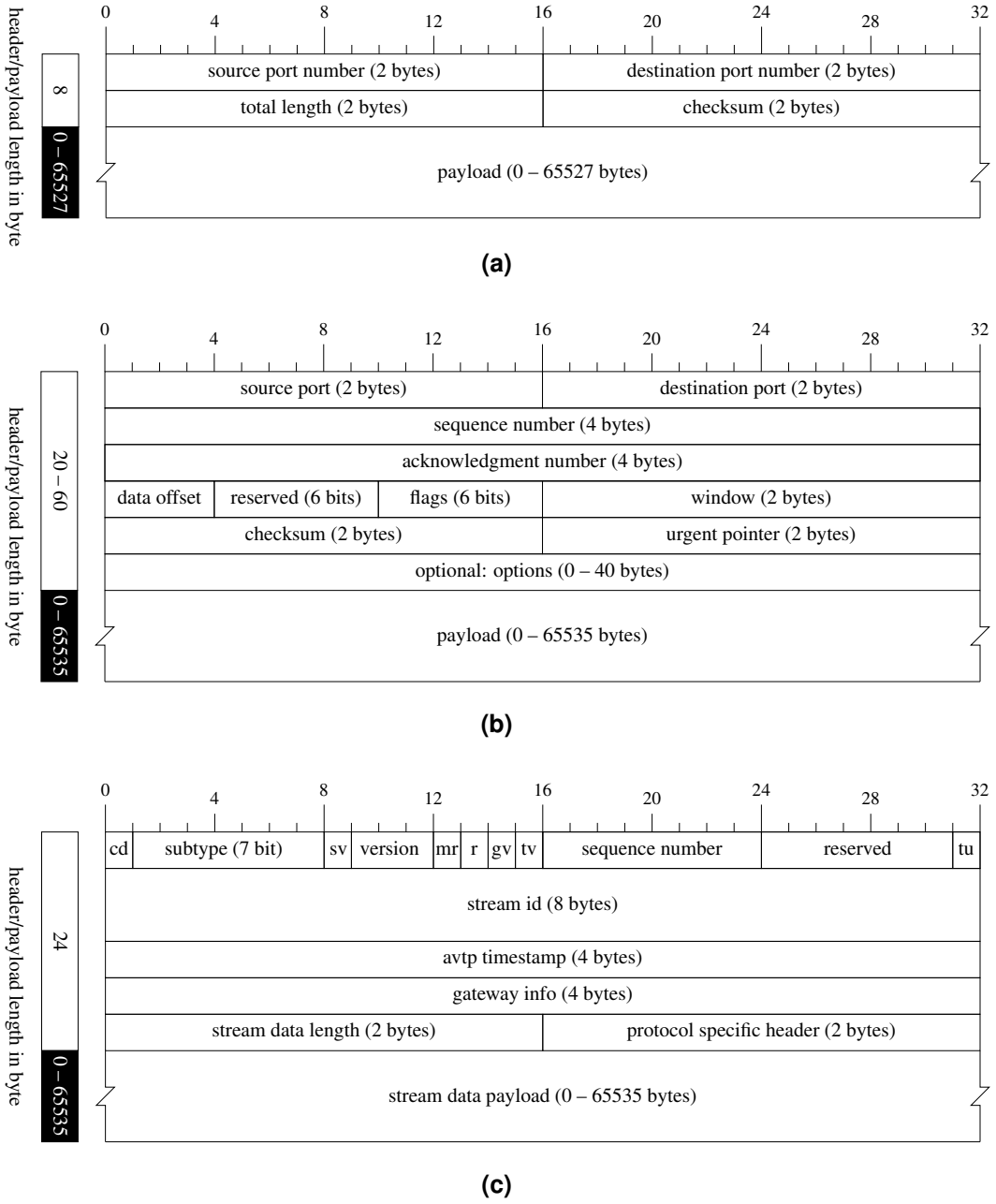


Figure 2.5.: Transport protocols headers of UDP, TCP, and IEEE 1722: (a) Shows the UDP frame format as defined in [Pos80]. The TCP frame format as specified in [Req81c] is presented in (b). Finally, (c) shows the media stream transport protocol frame format of IEEE 1722 as defined in [IEE11a].

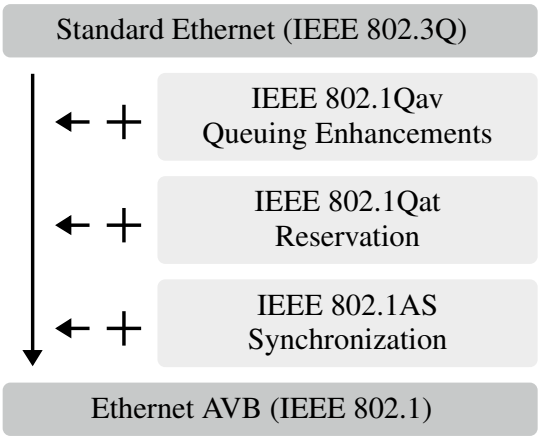


Figure 2.6.: Overview of Ethernet AVB. The standard extends the IEEE 802.3Q Ethernet standard by the three sub-standards IEEE 802.1Qav [IEE10c], IEEE 802.1Qat [IEE10d], and IEEE 802.1AS [IEE10b].

AVB extends the IEEE 802.3Q standard [IEE06] with three additional sub-standards as shown in Figure 2.6.

Queuing Enhancements (IEEE 802.1Qav) The standard utilizes methods described in IEEE 802.1Q [IEE06] to separate time-critical and non time-critical traffic into different traffic classes. Therefore, egress port buffers are separated into different queues, each allocated to a specific class. This ensures a separation of low priority traffic from high priority traffic. Moreover, all egress ports have a credit-based shaping mechanism to prevent a bursty behavior of packets on the wire.

Stream Reservation (IEEE 802.1Qat) The IEEE 802.1Qat [IEE10d] standard defines mechanisms for stream reservation within the network. An application endpoint can request a media stream by sending a reservation request. The reservation protocol will check, if the network load is able to handle the request. If it is possible, a positive response is sent and the necessary stream throughput is reserved at all stations between the transmitter and receiver of the requested stream.

Time Synchronization (IEEE 802.1AS) This standard describes the time synchronization within the network and is responsible for the precise time synchronization of the network nodes to a global time [IEE10b].

2.2. Analysis of Bit and Residual Error Rates

Based on the facts, that residual error estimations or exact calculations depend on the bit error probability of a given network link and that no bit error probabilities are available for automotive Ethernet products, a concept is developed in the following to determine the bit error probability and the residual error probability of a given automotive Ethernet link. In the following, the terms error and error rate are defined, the error detection concept is introduced to measure the bit and residual error rates, and the measured results are presented. The implementation of the concept uses an embedded system approach to enable vehicle measurements.

2.2.1. Fundamentals

An example of a bit error is presented in Figure 2.7. It shows two nodes interconnected with each other by a physical network connection. The transmitter t sends a number of bits represented by the binary data vector $\mathbf{d}$ to the receiver r. During the transmission, unplanned electromagnetic disturbances on the physical medium can lead to a misinterpretation of bits on the receiver side. This phenomena is described by a so-called *error vector* $\mathbf{e}$ which is applied to the initial data vector $\mathbf{d}$ using a XOR operation. A '1' at position i of the error vector $\mathbf{e}$ denotes an altered bit of the data vector $\mathbf{d}$ at position i during transmission. The resulting data vector $\mathbf{d'}$ on the receiver side is calculated by $\mathbf{d'} = \mathbf{d} \oplus \mathbf{e}$.

Definition 2.1 (Bit Error) A *bit error* (BE) is a bit that has been altered during transmission.

Definition 2.2 (Packet Error) A *Packet Error* (PE) occurs when one or more bits within certain packet have been altered during the transmission of this packet.

Definition 2.3 (Residual Packet Error) A *Residual Packet Error* (RPE) occurs when a packet passes error detection or error correction mechanisms even though the received or corrected packet contains at least one bit error.

Definition 2.4 (Bit Error Rate) The *Bit Error Rate* (BER) is the number of incorrectly received bits divided by the overall number of transmitted bits.

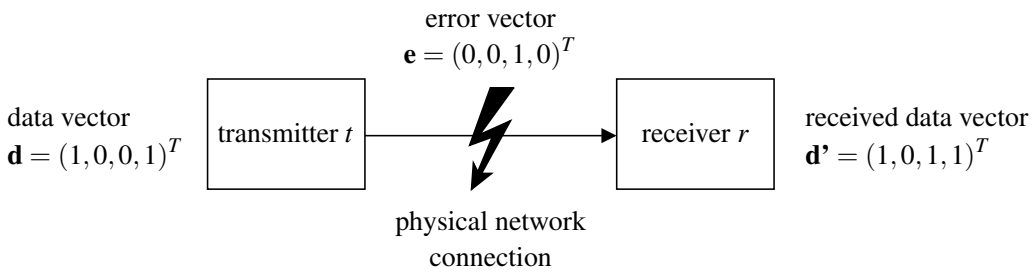


Figure 2.7.: Shown is the influence of a bit error **e** on a data vector **d**. In this example, a sequence of four bits is transmitted from the transmitter t to the receiver r and the bit at position three is altered.

$$BER = \frac{\text{number of } BEs}{\text{overall number of transmitted bits}} \tag{2.1}$$

Definition 2.5 (Packet Error Rate) The *Packet Error Rate* (PER) is the number of incorrectly received packets divided by the overall number of transmitted packets.

$$PER = \frac{\text{number of } PEs}{\text{overall number of transmitted packets}} \tag{2.2}$$

Definition 2.6 (Residual Error Rate) The *Residual Error Rate* (RER) is used with error detection or error correction mechanisms and denotes the number of packets which are accepted by the receiver even though the received or corrected packet contains at least one bit error.

$$RER = \frac{\text{number of } RPEs}{\text{overall number of transmitted packets}} \tag{2.3}$$

Definition 2.7 (Residual Error Probability) The *Residual Error Probability* (REP) P_{re} is the probability that an error during data transmission remains undetected. It depends on the bit error rate and bit error distribution.

$$P_{re} = \lim_{n_p \to \infty} \frac{\text{number of } RPEs}{\text{overall number of transmitted packets } n_p} \tag{2.4}$$

In Ethernet/IP systems, the residual error rate or probability typically depends on several nested error detection mechanisms which are mainly implemented as *Frame Check Sequences* (FCSs). In general, a FCS represents an extra field added to a frame or packet in a communication protocol for error detection and correction. Ethernet provides a CRC named IEEE 802.3 CRC-32 to detect bit errors. In addition, IP uses a checksum mechanism to detect bit errors within the IP header. Transport protocols like UDP or TCP again offer checksum mechanisms to protect the payload. Beside this, additional error detection mechanisms can be supported by the application layer. The residual error rates and probabilities can be based on different nested detection or correction mechanisms. For example, when looking at the residual error probability of Ethernet P_{re}^{Eth}, only the IEEE 802.3 CRC-32 protection is considered to detect errors. In case the IP checksum mechanism is used in addition to the Ethernet checksum, the residual error probability is denoted by P_{re}^{IP}. If the mechanisms from Ethernet, IP, and for example UDP are used, the residual error probability is denoted by P_{re}^{UDP}.

2.2.2. Related Work

The following research activities focus on the determination and improvement of the residual error probability P_{re}. An overview of different methods to exactly calculate or approximate the residual error probability is presented. In addition, research papers are introduced which calculate and compare the residual error probability of the automotive and non-automotive network technologies LIN, CAN, FlexRay, and Ethernet. Finally, design space exploration mechanisms are presented that try to find better generator polynomials for CRCs to improve the hamming distance to detect more error patterns. In addition, the behavior of nested error detection mechanisms is discussed.

In [MPSH07] and [BH07], different ways to determine the residual error probability P_{re} are summarized. Figure 2.8 shows the different simulation-based and analytical approaches to approximate or even exactly calculate the residual error probability for a given CRC polynomial p, bit error probability P_{bit}, and data length l. The *Monte Carlo Method* uses random samples to estimate P_{re} by the ratio of the number of undetectable error patterns to the number of overall samples. This method provides an approximated value for the residual error probability. An exact value can be calcu-

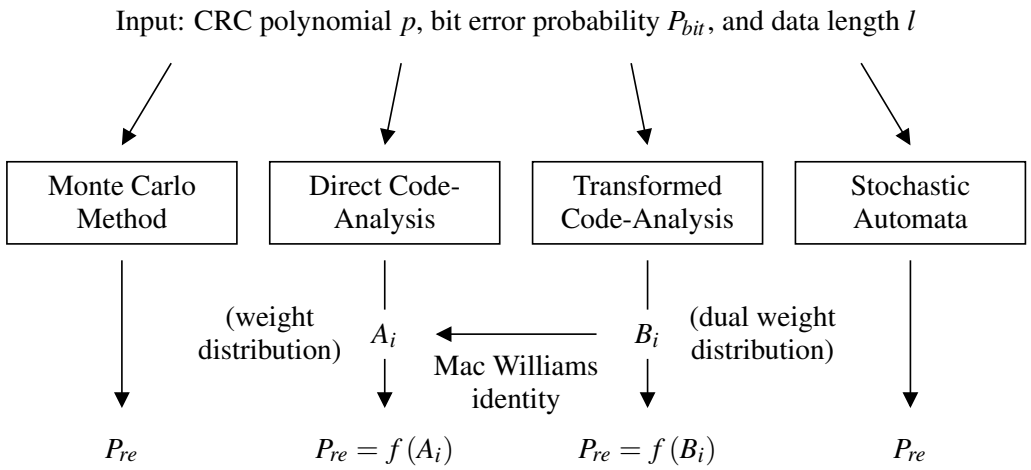


Figure 2.8.: Survey of the methods for the determination of the residual error probability P_{re} based on the CRC polynomial p, the bit error probability P_{bit}, and the data length l according to [MPSH07].

lated by the *Direct Code-Analysis*. In this case, all 2^l undetectable error vectors $\mathbf{e}$ have to be generated explicitly. The numbers A_i of those i erroneous bit vectors have to be counted $(A_1, A_2, ..., A_l)$. The residual error probability of a weight distribution A_i can now be calculated with the following equation:

$$P_{re} = \sum_{i=1}^{l} A_i \cdot P_{bit}^i \cdot (1 - P_{bit})^{l-i} \tag{2.5}$$

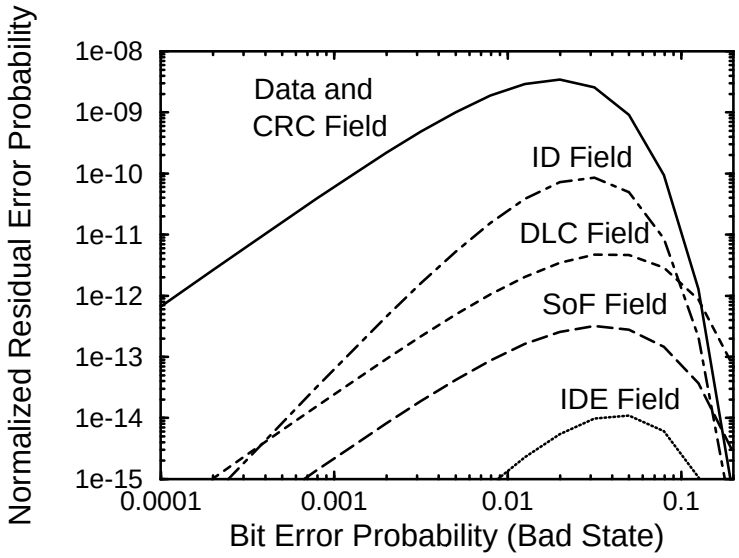
The calculation of all necessary error patterns leads to a complexity of $\mathrm{O}\left(2^l\right)$. Therefore, the computation is only feasible for short data lengths l. The *Transformed Code-Analysis* uses a much smaller set of patterns 2^r of the corresponding dual code. The weight distribution B_i of this code is determined and then P_{re} can be calculated directly. In addition, it is possible to use the *Mac Williams Identity* to get more precise results. Both ways sometimes lead to numerical problems and to inaccurate results. The *Stochastic Automata* method avoids such numerical problems [SM05]. The idea is to model the behavior of the *Linear Feedback Shift Register* (LFSR) which is typically used to implement the calculation of the CRC by a deterministic automaton. This automaton is extended to a stochastic variant by using the bit error probability P_{Bit}.

In summary, the exact calculation of the residual error probability is only feasible for short data lengths l and all calculations depend on the bit error probability P_{Bit} and

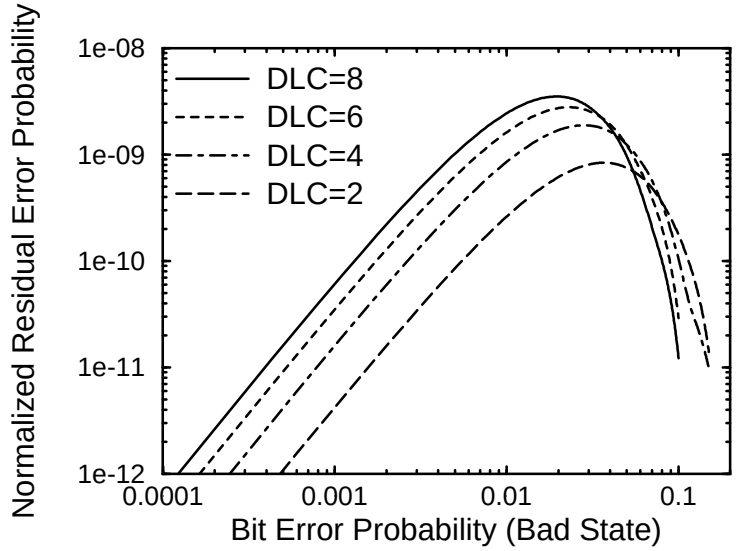
distribution. Therefore, it is necessary to know the bit error probability and distribution to give a statement about the robustness of given error detection or correction mechanisms. In this case, the robustness is the ability to detect or correct communication errors during transmission. The harder it is to create a residual error the more robust the algorithm is.

A detailed analysis of the error detection mechanisms of today's CAN technology is presented in [Cha94]. The author presents the different error detection mechanisms of CAN like the CRC mechanism and multiple plausibility checks to detect misbehavior. For the residual error probability calculation, a two-state symmetric binary channel model for the physical transmission media is used. A symmetric channel assumes a direction-independent bit error rate. Figure 2.9(a) highlights the contributions of the different protocol fields of the CAN protocol to the residual error probability depending on different bit error probabilities P_{bit}. The results of the impact of various packet lengths denoted with *Data Length Code* (DLC) to the residual error probability are shown in Figure 2.9(b). It is shown, that even high bit error probabilities of 10^{-4} lead to residual error probabilities of about 10^{-12}. In [RMRHS07], a comparison of the residual error probabilities of LIN, CAN, FlexRay, and Ethernet is presented. Figure 2.10 shows the results of the residual error probability calculations depending on different bit error probabilities P_{bit}. The Ethernet results are based on the minimum payload length of 42 bytes and show worse results compared to the residual error probabilities of CAN and FlexRay.

The finding of better generator polynomials for CRCs to detect more error vectors $\mathbf{e}$ is another research area. Several publications [Koo02], [KC04], and [RK06] from Koopman et al. exist which deal with the problem of finding better CRCs. The analysis are based on exhaustive searches of different k-bit CRC design spaces. First results present CRC-32 generator polynomials which support hamming distances up to 6 compared to the IEEE 802.3 CRC-32 hamming distance of 4 for maximum-length Ethernet messages. Further publications address the problem of finding better CRCs for embedded systems. Here, small k-bit CRCs are also considered as well as limited computational resources. Finally, in [MSM+08] different algorithms for the calculation of the residual error probability of several nested CRC combinations are developed. The result is, that the choice of different nested CRCs to achieve low residual error probabilities is application dependent and therefore there is no single best solution which fits all applications.



(a)



(b)

Figure 2.9.: The residual error probability P_{re} for different error positions within the packet and data length codes (DLCs). (a) Shows the contributions of the different error cases related to the protocol fields to the residual error probability P_{re} for standard CAN frames with 8 data bytes as functions of the bit error probability P_{bit}. (b) Highlights the impact of various data length codes to the residual error probability P_{re} as functions of the bit error probability P_{bit} [Cha94].

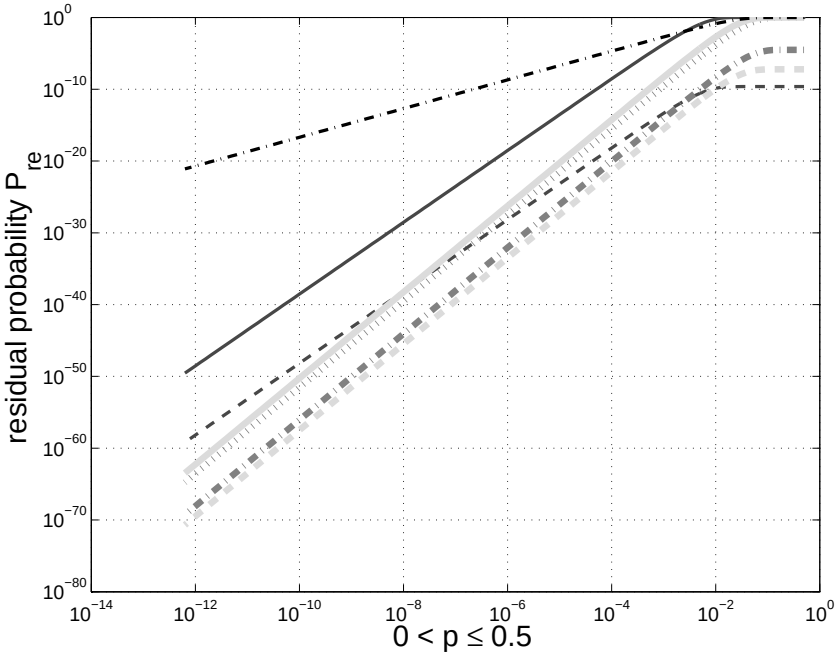


Figure 2.10.: Comparison of the residual error probabilities P_{re} of Ethernet and automotive network technologies depending on the bit error probability P_{bit} (denoted with p at the x-axis). The P_{re} (denoted with P at the y-axis) of CAN is computed for a payload length of 8 bytes (dotted curve: upper bound, thick dashed dotted curve: lower bound). The P_{re} of FlexRay is calculated for a payload length of 8 bytes (thick solid curve: upper bound, thick dashed curve: lower bound). The P_{re} of Ethernet is computed for a payload length of 42 bytes (thin solid curve: upper bound, thin dashed curve: lower bound). Finally, the P_{re} of LIN is calculated for a payload length of 7 bytes (thin dashed dotted curve: upper bound) [RMRHS07].

2.2.3. Error Detection Concept

Figure 2.11 shows the basic idea of an error detection mechanism for Ethernet networks. Two Ethernet-capable end points are used to establish an Ethernet link. The transmitter board sends predefined Ethernet-packets p_{send} via the Ethernet link to the receiver board. At the receiver side, each received packet p_{receive} is compared bit by bit with the expected packet p_{send} which is stored in memory. Therefore, the hardware CRC mechanism of the Ethernet controller has to be deactivated to enable the forwarding of faulty packets. Whenever an error is detected, a *packet error report*

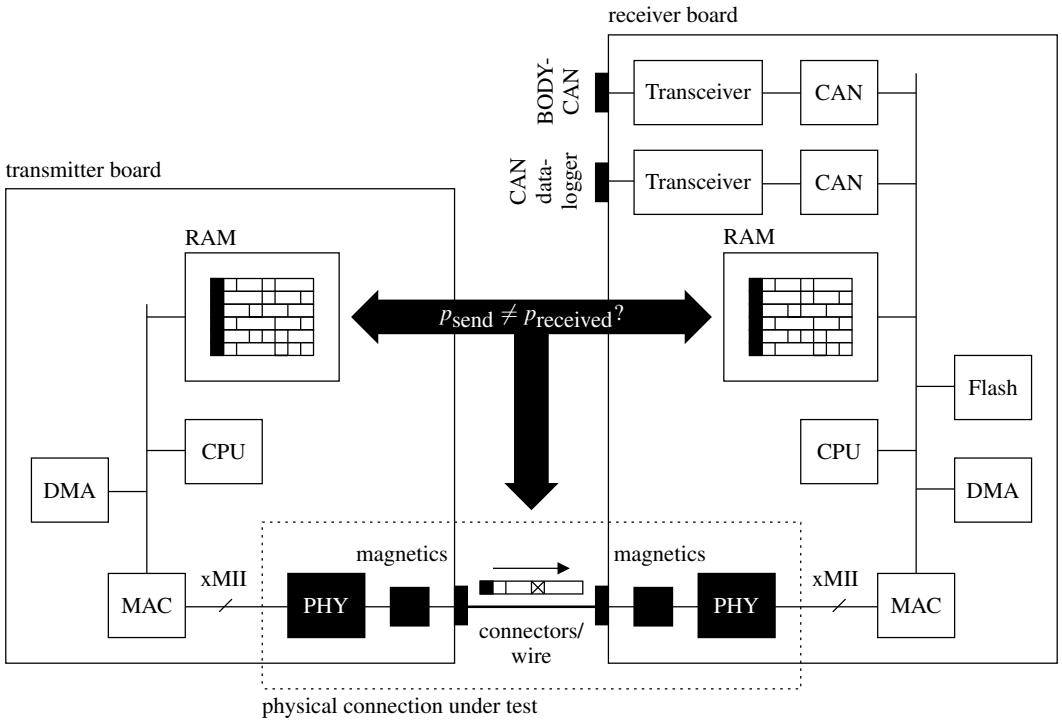


Figure 2.11.: Setup of a bit error and residual error measurement concept. The transmitter board sends predefined Ethernet-packets via the "physical connection under test" to the receiver board where the received packets are bit-wise analyzed to check for transmission errors.

is generated for the faulty packet and stored in the internal flash memory (first version) or in an external CAN data logger via the CAN interface (extended version). The CAN data logger contains a large hard disk drive to support long test runs over several days or even weeks which is important to get comprehensive results. In addition, a packet counter is introduced to detect packet loss or link cancellation and different PHY and Ethernet MAC parameters are recorded as well. Due to the limited amount of computational resources on the embedded design, it is necessary to use DMA mechanisms and offline evaluation to enhance the throughput of the Ethernet link. Offline evaluation means, that every faulty packet only generates a packet error report which contains all the information needed to reconstruct the whole faulty packet by an evaluation tool running on a computer. The error report also contains the current time and date which is obtained from the BODY-CAN of the vehicle or generated by a local timer when no vehicle BODY-CAN connection is available. After

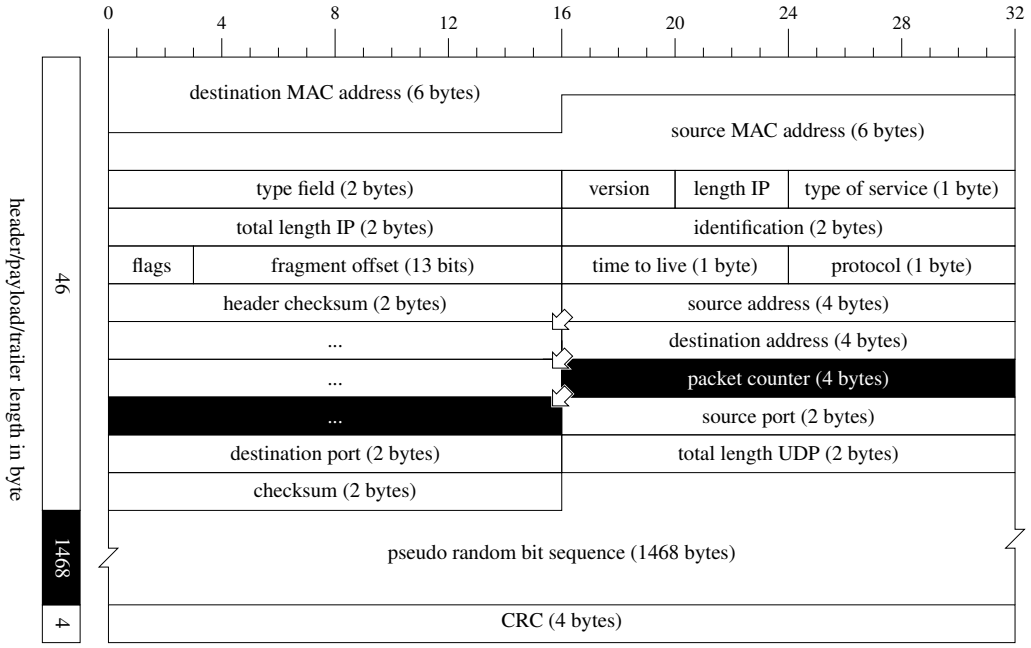


Figure 2.12.: The packet structure of the packets which are being transmitted from the transceiver board to the receiver board. The packet counter is stored within the IPv4 header extension to minimize the checksum recalculation overhead.

reconstructing the faulty packets, the tool calculates the bit error rate and distribution over the whole test run as well as the packet-based related values. In addition, the residual error probabilities of the nested error detection mechanisms are calculated according to Definition 2.7.

In this case, Ethernet, IP, and UDP are used to transmit pseudo random bit sequences from one node to the other. The complete structure of the packets which are being transmitted from the transceiver board to the receiver board is shown in Figure 2.12. The packet starts with the Ethernet header, followed by the IPv4 and UDP headers. The payload consists of a *pseudo random bit sequence* with a length of 1468 bytes to create an Ethernet packet with the maximum transmission unit of Ethernet of 1518 bytes. Finally, the Ethernet CRC is added. All packet fields can be statically preconfigured, except for the packet counter which is incremented by 1 with each packet. The packet counter is placed in the header extension of IPv4 to minimize the recalculation overhead of the error detection mechanisms which have to be updated. In this case, only the IP header checksum has to be recalculated.

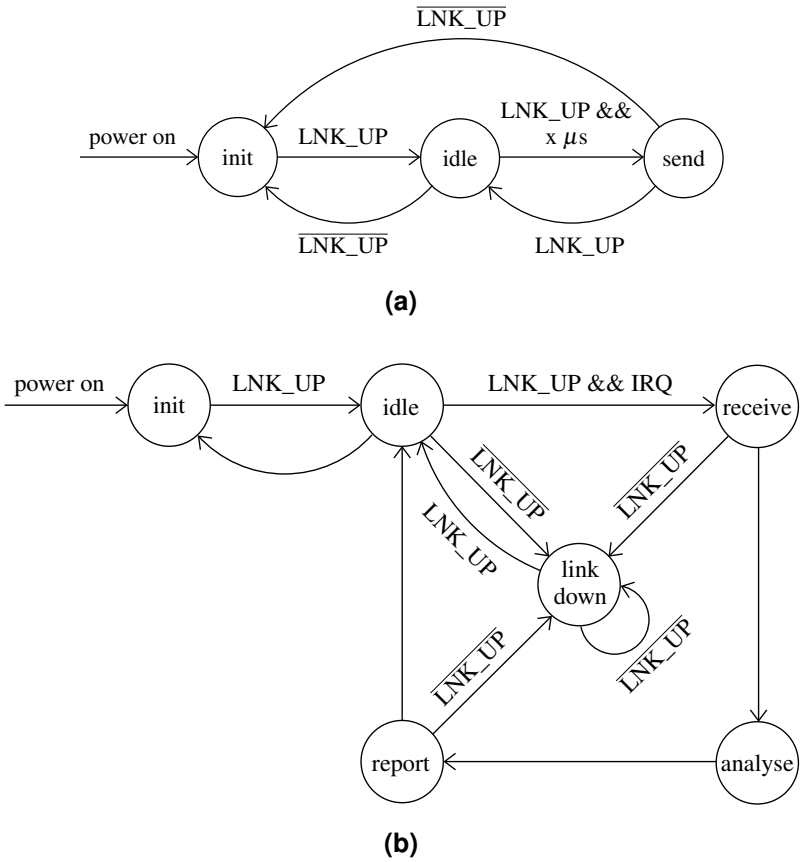


Figure 2.13.: (a) Shown is the functionality of the software of the transceiver board which continuously sends Ethernet-packets to the receiver board using a finite state machine model. (b) Described is the more complex behavior of the receiver board, also by a finite state machine. After receiving a packet, it is analyzed and an error report will be created if a transmission error has occurred.

The functionality of the software on the two boards is presented in Figure 2.13 by a finite state machine each. Figure 2.13(a) describes the software structure of the transceiver board. After initialization and link-up, the continuous transmission of the Ethernet-packets starts. If a link loss occurs, the board will return into the idle state until the Ethernet connection is reestablished. The behavior of the receiver board is shown in Figure 2.13(b). After initialization and link-up again, the board is waiting for an incoming packet which is signaled by an interrupt from the Ethernet controller. The received packet is then analyzed by performing a bit-wise comparison. If one or more bits are altered during the transmission, an error report is created by the software

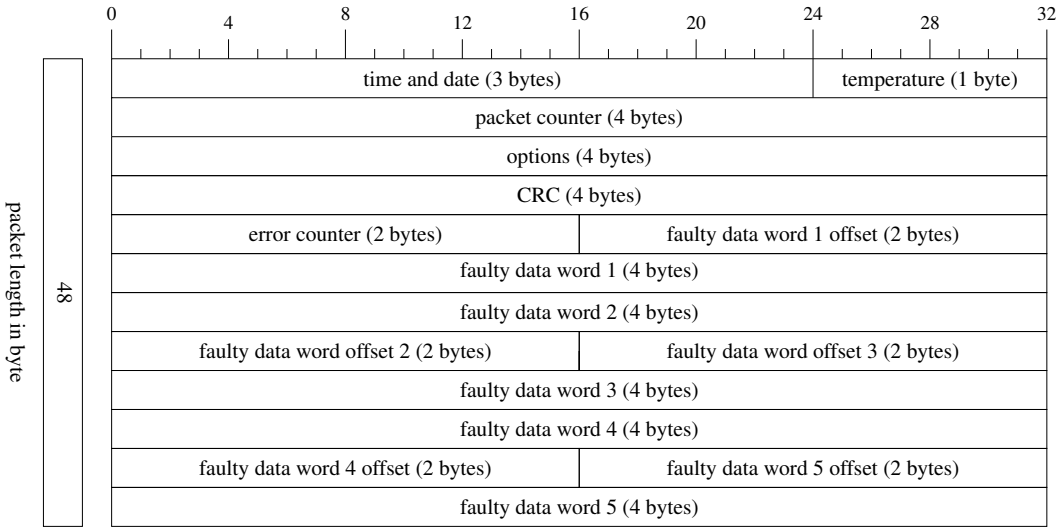


Figure 2.14.: The data structure of the error reports.

to record the packet error. After that, the receiver board resumes to the idle state and waits for the next incoming packet.

The structure of the error report depends on the storage mechanism. In the first instance, an on-board flash memory is utilized to store the results. The used data structure is presented in Figure 2.14 and starts with the environmental parameters *time*, *date*, and *temperature* followed by the current *packet counter*, different *options*, the *CRC* of the received packet, and the current *error counter*. The remaining fields store the different detected bit errors. Whenever a bit error occurs, the whole *data word* is stored together with the *offset* of the data word relative to the beginning of the packet. The flash-based variant supports up to five data words per packet. The disadvantages of this approach are the limited size of the error report data structure due to the limited size of the on-board flash which is typically about a few megabytes and the problem of writing the results from the board to the computer. In the second version, an external memory is used to store the error reports. Therefore, a second CAN interface was connected to a CAN data logger device. This device records all incoming CAN frames onto a hard disk drive with hundreds of gigabytes of space. The transfer of the data is performed by connecting an Ethernet cable to the data logger. This simplifies the read out and supports additional functions like automated backups and precise timing information about the stored data.

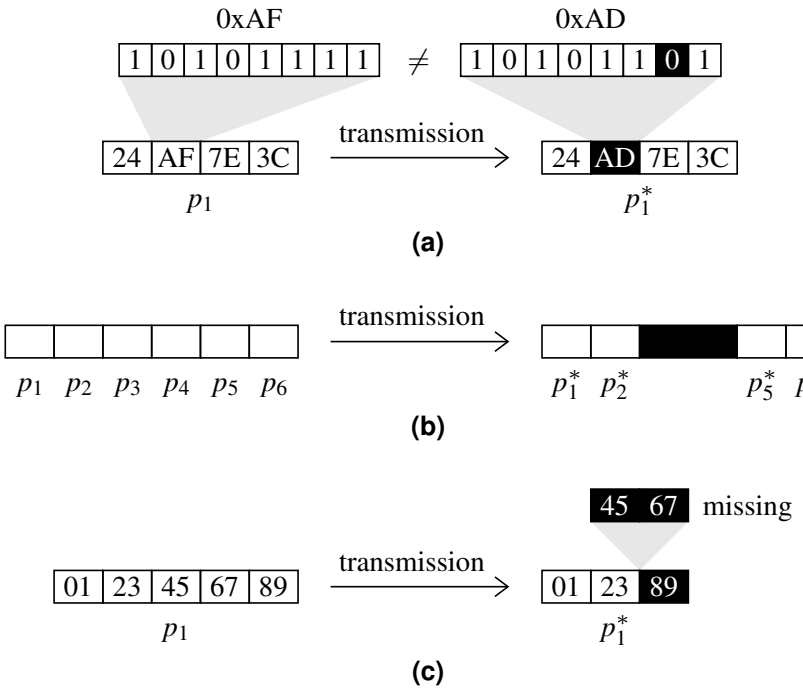


Figure 2.15.: Different observed error types: (a) Shows a typical bit error where one or more bits altered during transmission. (b) Shows the absence of one or more packets at the receiver side. (c) Shows the loss of one or more data words within a packet.

By assessing the stored error reports, the bit error rate and bit error distribution can be determined. In addition, it is possible to calculate the distribution of the number of faulty bits per packet. At packet-level, the packet error rate and the residual error rates according to the applied error detection mechanisms can be determined.

2.2.4. Error Type Classification

The measurements in the EMC chamber and vehicle measurements which are presented in the following sections led to three different error types. These error types are described in Figure 2.15. Figure 2.15(a) shows a typical bit error where one or more bits are altered during transmission. In this example, one bit altered during the transmission of packet p_1 to the receiver. The second error type describes the complete absence of one or more complete packets and is shown in Figure 2.15(b). The example shows six packets p_1 to p_6 which are transmitted to the receiver. At the receiver side, only the packets p_1, p_2, p_5, and p_6 arrived. The packets p_3 and p_4 were

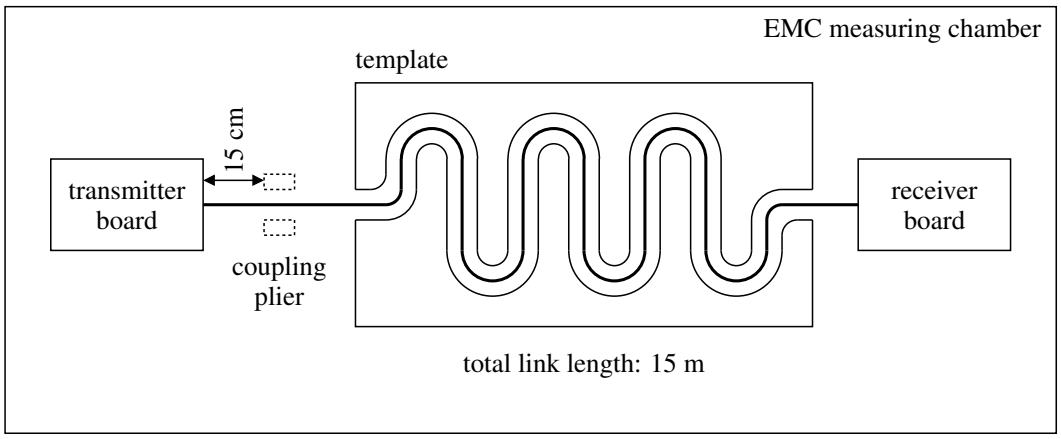


Figure 2.16.: Shows the setup of the BCI measurements within the EMC measuring chamber.

not received. One reason could be a loss of the Ethernet link. In such a case, a huge number of packets will get lost. The last observed error type represents the loss of data words within a packet and is shown in Figure 2.15(c). In this example, a packet p_1 consisting of 5 bytes was transmitted to the receiver. But the received packet p_1^* only contains 3 bytes. The third and fourth byte are missing at the receiver side.

2.2.5. Bulk Current Injection Results

This section presents the *Bulk Current Injection* (BCI) measurement results for a given Ethernet implementation. The used test setup is shown in Figure 2.16. All measurements were done in an *Electromagnetic Compatibility* (EMC) measuring chamber with one transmitter and one receiver board. Both boards had a metal housing and were interconnected via a twisted pair cable with a length of 590.55 inches (15 meters). The installation path was fixed by a template to ensure minimum installation spaces and to guarantee the reproducibility of the measurement. The coupling plier was installed in a distance of 5.91 inches (15 centimeters) from the transmitter board. All results are average values based on six consecutive measurements.

Figure 2.17 shows the average packet error of “error type a” depending on the frequency and the maximum interference current. L2K1/L1K3 and L1K2 denote the OEM-specific limiting values for non-safety-critical and safety-critical applications depending on the frequency and the coupled interference current. The dotted line

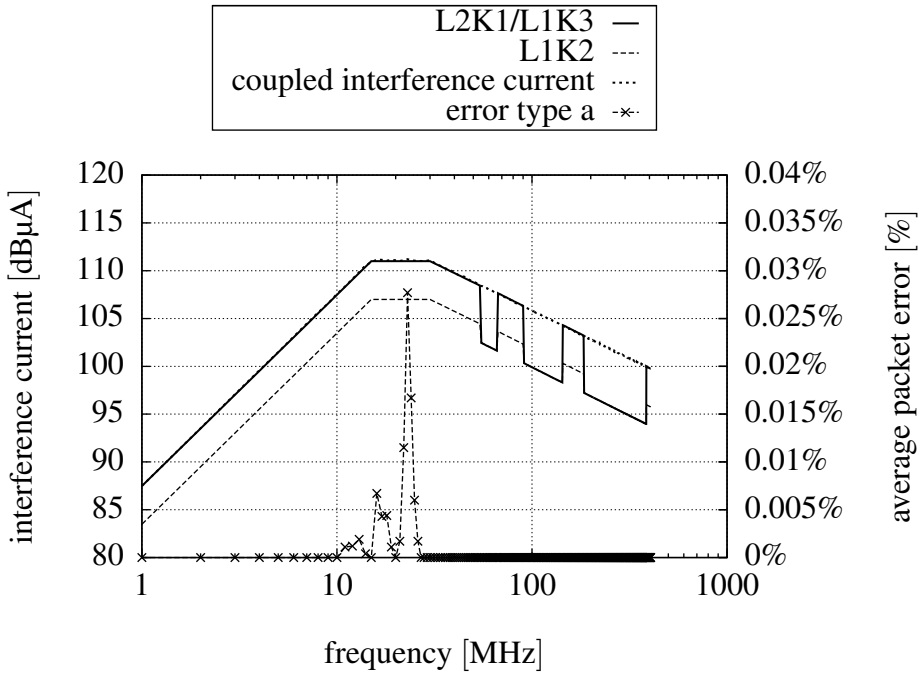


Figure 2.17.: The average values of the occurred packet errors of 'error type a' depending on the frequency and the maximum coupled interference current.

represents the applied current by the coupling plier during the measurement. Injected currents below the limiting values must not lead to communication errors. In this test setup, the frequency range of about 10 MHz to 40 MHz lead to several packet errors based on the altering of one or more bits.

A detailed bit-level view of the occurred packet errors is presented in Figure 2.18. Figure 2.18(a) shows the distribution of the number of bit errors per packet. With an IEEE 802.3 CRC-32 hamming distance of 4 for maximum length Ethernet packets, up to 3 bit errors can be reliably detected. With regard to this measurement, about 40 percent of all faulty packets have less than or equal to 3 bit errors and can thus be detected. For all packets with more than 3 bit errors, there is no guarantee that the checksum will recognize the faulty packet. But the offline evaluation showed, that all faulty packets could be detected by the IEEE 802.3 CRC-32. In this case, the additional use of the IP and UPD error detection mechanisms was not necessary. Figure 2.18(b) presents the general distribution of the bit error lengths. For example, a bit error length of 3 stands for 3 consecutive altered bits. Most of the measured

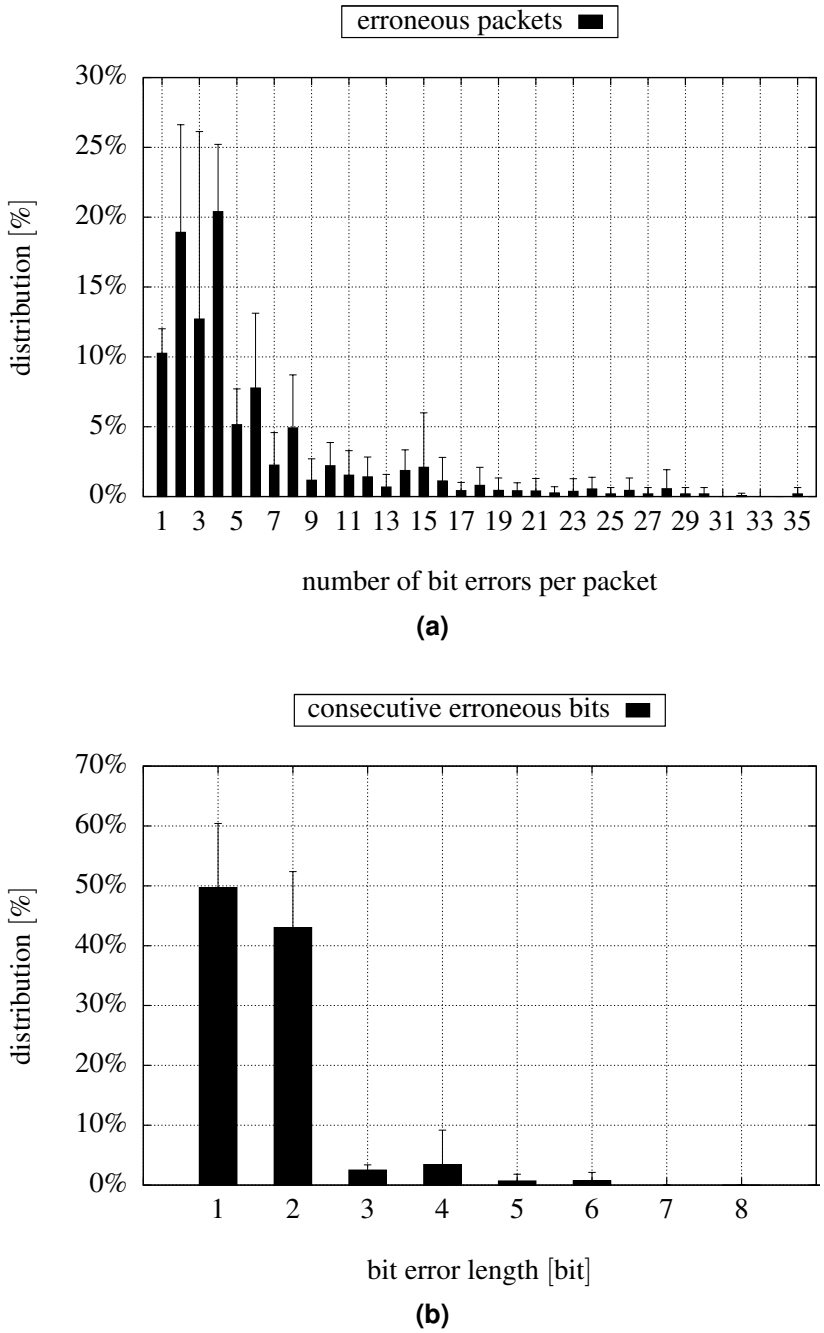


Figure 2.18.: (a) Shows the distribution of the number of bit errors per packet. (b) Presents the general distribution of the bit error lengths independent of the packet structure.

bit errors are single bit errors followed by two bit errors and the maximum measured bit error length was 6 bits. Note, that the bit error lengths are independent from the packet structure.

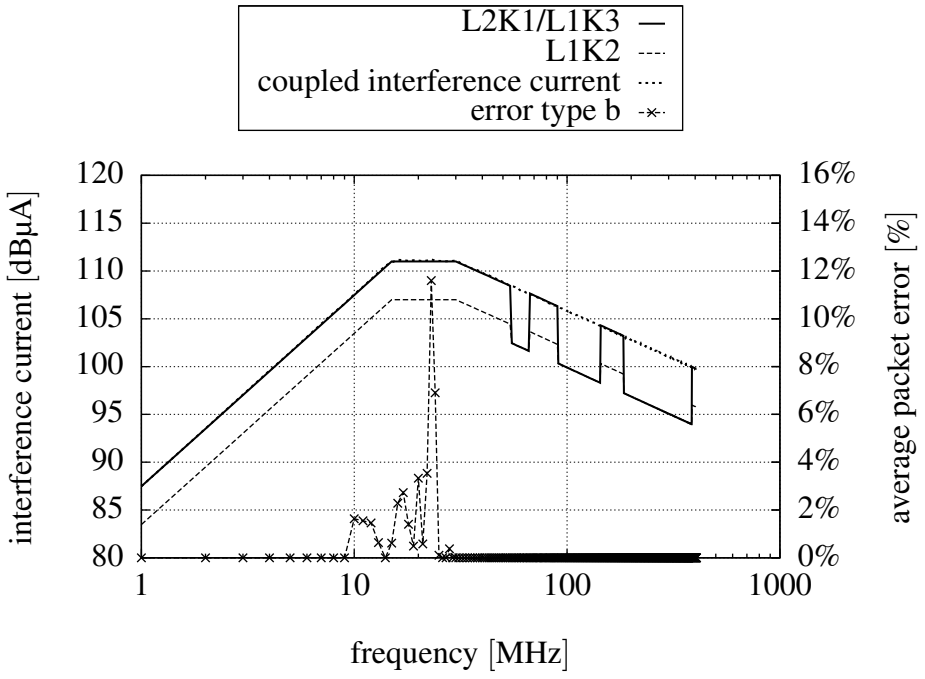
The results of the average packet error of "error type b" are presented in Figure 2.19(a) and showed much more occurred packet errors. The higher rates are based on the fact that during an error of type b multiple packets were discarded. In this cases, the difference between the packet counters of two successive received packets was greater than one. One reason could be link losses during the test run. Figure 2.19(b) shows the distribution of the number of consecutive lost packets for different interval lengths. For example, the interval length of $[500;1000]$ contains all occurred packets losses of 500 to 1000 successive packets. The maximum number of consecutively discarded packets was in an interval between $12,000$ and $12,500$ packets. Most errors implied packet losses of at most 500 packets. For example, the transmission time of 2000 packets at rate of 60 Mbit/s is about 500 ms on an 100 Mbit/s Ethernet link. This could be an indication for a link error.

Figure 2.20 shows the measurement results of "error type c". The percentage number is relatively low compared to the preceding error types but the errors occurred at the same frequencies.

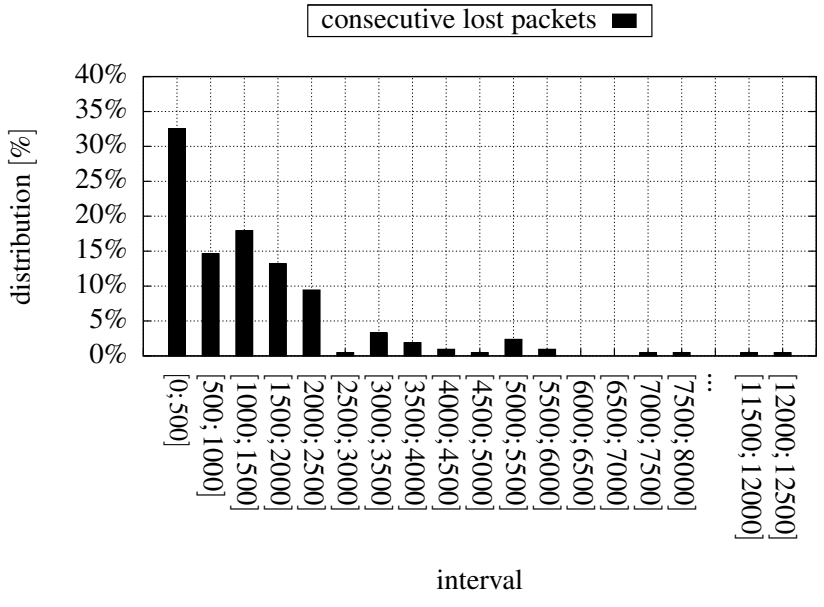
In summary, most of the errors were consecutive packet losts of "error type b" and occurred between a frequency range of 10 MHz to 40 MHz. The number of sent packets was about $3 \cdot 10^7$ packets. This resulted in a number of $3.6 \cdot 10^{11}$ sent bits. In total, $295,830$ packets and $6,963$ bits were faulty. The number of undetected errors was 0. This means, that no faulty packet was forwarded to the receiving application. However, packet loss could not be detected by higher layer applications due to the use of UDP. Reliable transport protocols like TCP can handle retransmissions to avoid packet loss in the case of errors.

2.2.6. Vehicle Integration and Results

A new Mercedes Benz E-class (type series 212) was adapted to perform the previously described vehicle measurements in situ. Figure 2.21 shows the wiring harness of the three test setups. All test setups consisted of a transmitter board F and a receiver board R. In the first test setup, for example, the transceiver board F1 was placed close to the engine to check if the engine could cause any problems. The cabling was an



(a)



(b)

Figure 2.19.: (a) Shows the average values of the occurred packet errors of 'error type b' depending on the frequency and the maximum coupled interference current. (b) Presents the distribution of the number of consecutive lost packets for different interval lengths.

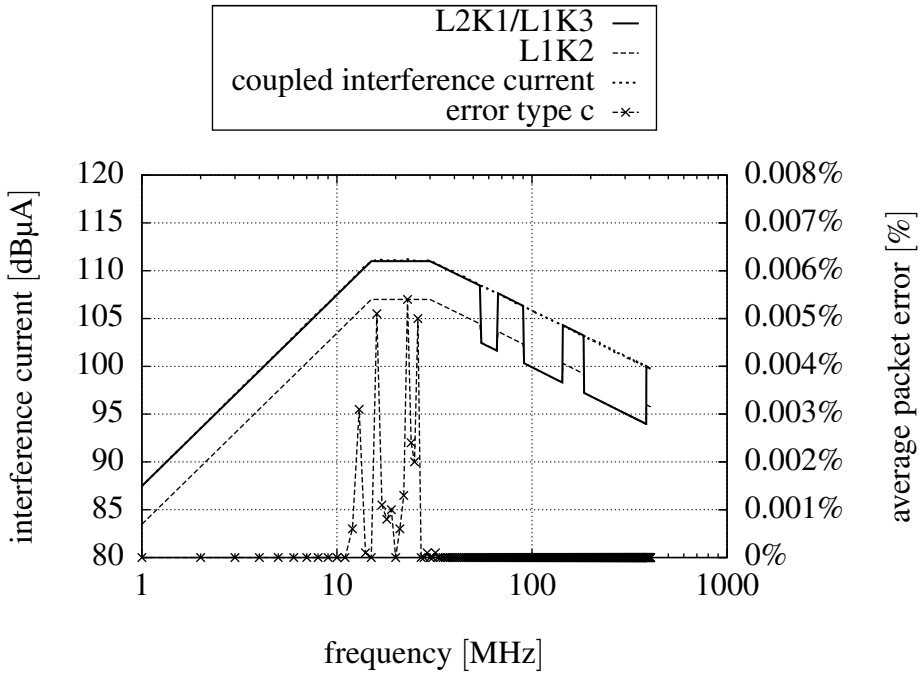


Figure 2.20.: The average values of the occurred packet errors of 'error type c' depending on the frequency and the maximum coupled interference current.

Unshielded Twisted Pair (UTP) cable with standard MQS connectors and the boards were housed in a plastic package. The receiver boards R1, R2, and, R3 were placed in the trunk of the car. After the car was started, the transceiver board started to transmit packets. The receiver board then received the packets, analyzed them and created error reports for faulty packets.

The following results presented in Figure 2.22 of "error type a" are based on the engine test setup. Figure 2.22(a) shows the distribution of the number of bit errors per packet. As described at the BCI measurement results, up to 3 bit errors can reliably be detected by the Ethernet CRC-32. The offline evaluation also showed, that there were no errors during the whole vehicle measurements which led to undetected errors. This means, that all occurred errors were detected by the unmodified checksum mechanism (CRC) of Ethernet. Figure 2.22(b) presents the general distribution of the bit error lengths independent of the packet structure. Most of the bit errors were single bit or two bit errors. The maximum bit error length was 15.

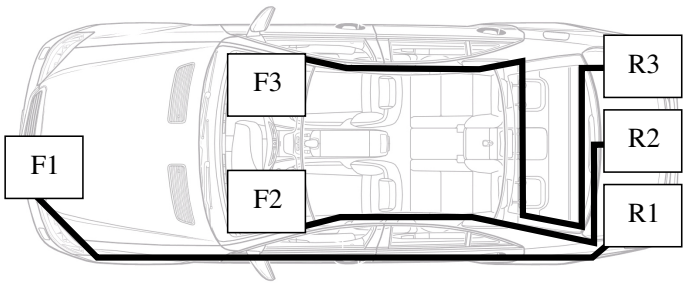


Figure 2.21.: The wiring harness of the three test setups.

During the whole vehicle measurements, a number of $8.3 \cdot 10^8$ packets was sent. This resulted in about 10^{13} bits. The number of faulty packets was $2,115$ and the number of faulty bits was $3,375$ bits. There were no undetected faulty packets during the whole test runs. The resulting bit error rate P_{bit} was $3.35 \cdot 10^{-10}$ and the packet error rate P_{packet} was $2.52 \cdot 10^{-6}$. The residual error rate was 0.

In summary, all measurements showed that no residual packet errors occurred. But the evaluation of the results revealed that there were bit errors greater than 3 per packet which could lead to residual packet errors. To avoid this, additional error detection mechanism are necessary. In general, in the automotive section the so called *Automotive Safety Integrity Level* (ASIL) [ISO11c] classification is used to classify safety critical functions. Each level has different requirements concerning the communication error rates. Depending on the ASIL level, additional error detection mechanisms are necessary. Note that these measurements were done with preliminary hardware. With further improvements and hardware changes, these evaluations have to be repeated.

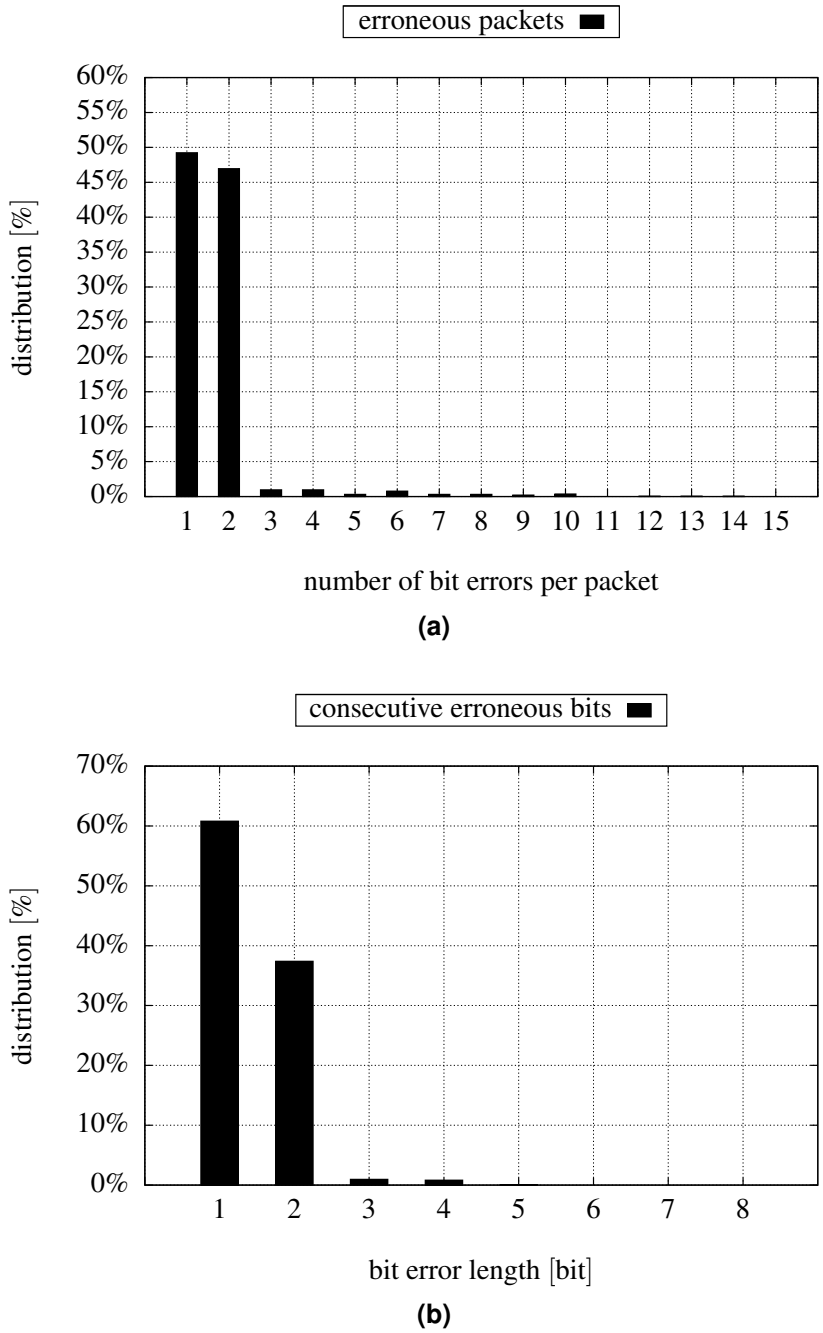


Figure 2.22.: The results of the vehicle measurements. (a) Shows the distribution of the number of bit errors per packet. (b) Presents the general distribution of the bit error lengths independent from the packet structure.

2.3. Accuracy of Packet-based Time Synchronization Algorithms

In order to fulfill automotive-specific requirements like reliable synchronization of distributed embedded systems under varying climate conditions, a number of verifications as described in the following have to be done. This section deals with the robustness of the packet-based synchronization algorithm specified in IEEE 802.1AS. Different setups are evaluated within climate chambers to test the synchronization accuracy under varying temperature conditions from $-10\,°C$ ($+14\,F$) to $+70\,°C$ ($+158\,K$) with different speeds in changes. In the following, an evaluation concept is introduced together with an IEEE 802.1AS test setup and the extensive measurements are presented.

2.3.1. Fundamentals

Synchronization algorithms are used to establish a common time base within a distributed system [IEE10b]. For example, when having four wheel-speed sensors, it is necessary to know the exact point in time where each of the values was sampled. In the case of object recognition based on distributed cameras around the car, it is also essential to know the exact capture time of each frame to ensure that detected objects from different cameras are time synchronized. Each node within a distributed system has its own free running local clock. The task of a time synchronization algorithm is to adjust the offset and drift of the different free running clocks to a common time base. This is typically done by the continuously exchange of *synchronization messages* (sync-messages) with a defined structure containing timestamps. The maximum relative offset between any two clocks in the synchronized system defines the synchronization *accuracy*. The following sections describe time synchronization algorithms and techniques in physical-shared and switched communication networks.

2.3.1.1. Synchronization in Physical-shared Networks

Todays common automotive communication technologies are CAN and FlexRay. Both technologies are based on a physical-shared medium and support unicast, multicast, and broadcast message exchange at the physical level. CAN uses an event-

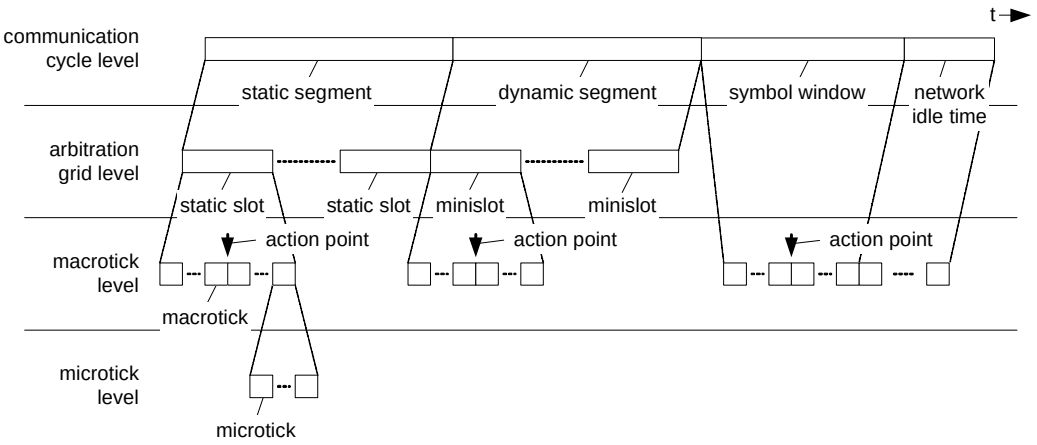


Figure 2.23.: The FlexRay communication cycle with its timing hierarchy. The communication cycle is divided in the four parts static and dynamic segment, symbol window, and idle time. Every part consists of a defined number of so-called microticks which are the smallest time duration within FlexRay [Fle10].

driven message delivery approach and does not initially support time synchronization. Therefore, additional algorithms are needed to support time synchronization. In [RNP04], different approaches are compared and evaluated. Most of the algorithms use high-priority multicast messages containing timestamps of the network nodes to calculate time offsets and drifts. FlexRay already provides a synchronized network infrastructure due to the use of an underlying time-triggered communication protocol. It uses a TDMA scheme and divides the bus access into so-called cycles, segments, slots, macroticks, and microticks. Figure 2.23 shows an overview of the bus access scheme of FlexRay. The network time synchronization is done by sending multicast sync-messages in predefined static slots. When receiving sync-messages, nodes can calculate the time offsets and drifts by knowing the predefined communication cycle duration and the offsets of each sync message related to the start of a communication cycle. Due to the support of a 64 communication cycle network synchronized counter, an additional message-based synchronization algorithm can be used to enable an application synchronized timer. Different software scheduling methods are available [Vec09] to adapt the FlexRay schedule to an operating system process schedule and vice versa. This enables full time synchronized distributed application tasks.

2.3.1.2. Time Synchronization in Switched Networks

Switched networks like Ethernet are not based on one single physical medium. Typically, single point-to-point connections are used to interconnect two nodes and larger networks are build using switches, which interconnect more than two nodes. Moreover, multicast and broadcast message exchange have to be supported by the switches and this is not done by receiving the same message on different nodes connected to the same physical media as in the case of physically shared networks. Switches duplicate multicast and broadcast messages to every outgoing port and enqueue them in the corresponding egress queues. Thus, additional latencies can be caused by the different egress queues which will lead to higher differences in the latencies on the receiving nodes for one multicast or broadcast message. That differing behavior has a significant influence on the time synchronization mechanisms. In the following, the three popular packet-based synchronization standards based on the *Precision Time Protocol* (PTP) [IEE02] are presented.

IEEE 1588-2002 This first PTP standard was published in 2002 by the IEEE and was designed to achieve clock synchronization accuracies in the sub-microsecond range. It synchronizes distributed local clocks referred as slave clocks with a reference clock (master clock). Figure 2.24 shows the principle message flow between a timing master node and a slave device to exchange all necessary timestamps to calculate the time offset and drift to the master clock. It starts with the dispatch of a *sync* message from the master to the slave device. The point in time when the message leaves the device is denoted with t_1 and is immediately forwarded by a *follow-up* message to the slave. The slave uses its local clock to timestamp the receipt of the sync message with t_2. After that, a *delay request* message is sent at the time t_3 from the slave to the master device and then the master responds with a *delay response* message containing the timestamp t_4 to the slave. The calculation of the packet delay, clock offset, and propagation delay can then be done using the timestamps t_1, t_2, t_3, and t_4. Equations (2.6) and (2.7) show the calculation of the packet delay and clock offset. Since the clocks in the network can drift independently and continuously, the synchronization algorithm is executed periodically every t_s microseconds.

$$t_{delay} = \frac{(t_1 - t_2) + (t_4 - t_3)}{2} \tag{2.6}$$

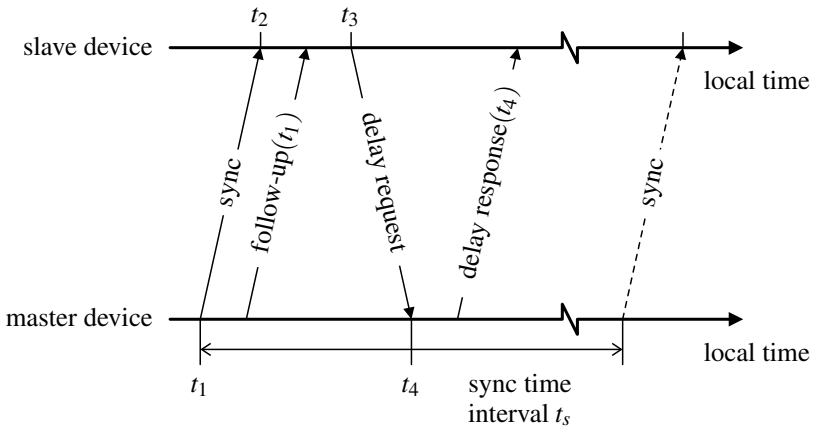


Figure 2.24.: Shown is the PTP message flow between the master clock device and the slave device. The messages *sync*, *follow-up*, *delay request*, and *delay response* are used to record and exchange the timestamps t_1, t_2, t_3, and t_4. These four timestamps are the basis for calculating the packet delay t_{delay} and the clock offset t_{clock}.

$$t_{clock} = \frac{(t_1 - t_2) - (t_4 - t_3)}{2} \tag{2.7}$$

IEEE 1588-2008 Version 2 of the initial standard released in 2002 contains a list of improvements to enhance the usability and the precision in large networks [VK08]. Among others, shorter synchronization frames were defined to reduce communication load and the use of transparent clocks beside ordinary clocks avoids error propagation in cascaded networks. Transparent clocks allow the correction of the propagation delay by adding a correction field to the sync message which contains the residence time of a packet within a switch for example. In addition, the configuration of the protocol is much more flexible by defining *profiles*.

IEEE 802.1AS The IEEE 802.1AS standard is an IEEE 1588-2008 profile. It defines that the propagation delays between two nodes are measured using the peer-delay mechanism, allows only ordinary clocks or peer to peer transparent clocks, and uses two-step clocks. In addition, the *Best Master Clock Algorithm* (BMCA) determines the *Clock Master* of an AVB network and provides the reference clock for the networked devices. The BMCA finds the best clock in the network and constructs a

time synchronization spanning tree with the reference clock as the root. As part of constructing the time synchronization spanning tree, each port of the synchronized system is assigned a *port role*. The port which is closer to the Clock Master compared to its direct communication path port gets the master role, the other gets the slave role. To determine the best clock in the system, each clock has so-called *stratum number* as defined in IEEE 1588. The clock with the lowest stratum number is chosen as the Clock Master in the network. If the number of clocks with the lowest stratum number is greater than 1, then the clock with the largest MAC address is chosen. The specified worst case accuracy between two end nodes is less than one microsecond.

2.3.2. Related Work

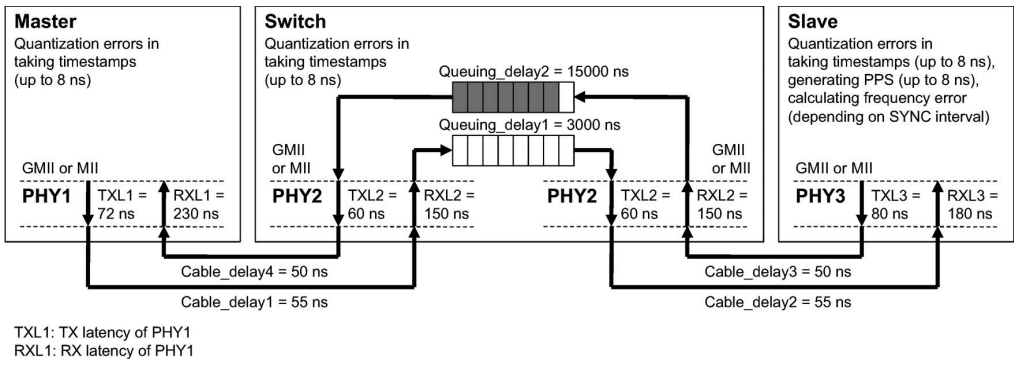
A lot of research has been done to investigate the synchronization accuracy of distributed real-time networks running PTP with ordinary and transparent clocks representing the basis for IEEE 802.1AS. In [HJ10], the authors introduce a practical implementation of IEEE 1588-2008 together with a test setup to verify the submicrosecond accuracy of their implementation. Figure 2.25(a) gives an overview of their relevant timing delays for a routing path from the master clock to the slave clock containing one switch in between. The sample propagation delay d in each direction is calculated by Equations (2.8) and (2.9):

$$d_{m\to s} = TXL1 + CD1 + RXL2 + QD1 + TXL2 + CD2 + RXL3 = 3572\,ns \quad (2.8)$$

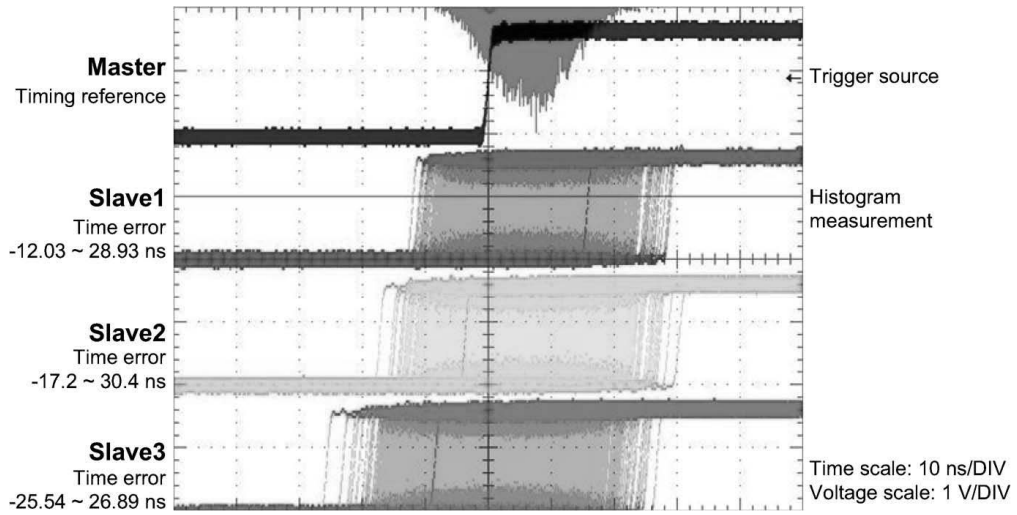
$$d_{s\to m} = TXL3 + CD3 + RXL2 + QD2 + TXL2 + CD4 + RXL1 = 15620\,ns \quad (2.9)$$

Figure 2.25(b) shows the measurement results based on a network with four nodes and three switches placed in series. Each switch is connected to one slave node running a slave clock and the first switch is additionally connected to the master node running the master clock. The results show an upper-bound synchronization accuracy of about 27 ns over three switches.

Another implementation is introduced and evaluated by [EFH+08]. They present a *spider transparent clock* which corrects the internal queuing jitter and asymmetry introduced by bridges and routers. The concept works transparent to IEEE 1588 and



(a)



(b)

Figure 2.25.: The relevant timing delays for a packet which is sent from the master to the slave are presented in (a). Delays are caused by the PHYs, the wires, and the queuing delays within the switch. (b) Highlights some clock synchronization measurements. It shows the jitter for different slave nodes for a given time interval [HJ10].

leads to a synchronization accuracy of about 80 ns for two nodes and one switch in between. In [FFM+07], the authors implemented and compared three synchronization techniques. The first one uses an external dedicated 1-PPS signal which led to the best results with an average accuracy of 15 ns. The other two methods are IEEE 1588 and PCTP (PROFINET IO). They present results for different cabling methods with an upper bound synchronization accuracy of a few hundred nanoseconds. Additional

implementations, test setups, and evaluations are presented in [KÖ7, BGNV09]. In summary, the authors presented different implementations and test setups from simple networks with two end devices and one switch to more complex networks with several endpoints and a multitude of switches in between. Moreover, they created additional data traffic to experience larger queuing delays within the network elements. The presented synchronization error measurements are compared to ordinary switches and have a wide range varying from a few nanoseconds to several hundred nanoseconds.

A fundamental test configuration of an IEEE 802.1AS network profile based on IEEE 1588-2008 for performance analysis is described in [TG08]. First results based on simulations for an optimal IEEE 802.1AS network are presented in [GGT09]. Figure 2.26 shows the *Maximum Time Interval Error* (MTIE) for different observation intervals together with requirements for uncompressed standard and high definition video (SDTV and HDTV), consumer and professional audio, and several cellular base station technologies. The dashed lines in the middle of the plot which are very close together show the results for the MTIE for one to seven hops with and without background traffic. In the case of professional audio, the jitter requirements can be met for the observation intervals ranging from 0.001 s to 10 s.

All investigations have in common that they do not consider any thermal influences like very low or high temperatures and different speeds in temperature changes. However, this is an essential requirement for deploying a communication system in automotive applications. Therefore, car manufacturers have company standards, see [Mer09], which describe special tests in extreme temperature conditions or thermal shock which have to be performed and passed.

2.3.3. Test Environment

The physical behavior of crystals, the clock adjustment algorithm, the sample or media clock distribution, and the sample clock jitter are explained in this section.

Crystals and Clock Adjustment For the clock synchronization accuracy, the physical properties of the crystal are an essential parameter. The major influence on the frequency itself is the temperature. Operation under varying temperature conditions causes the crystal to vary its output frequency. Typically, they will resonate

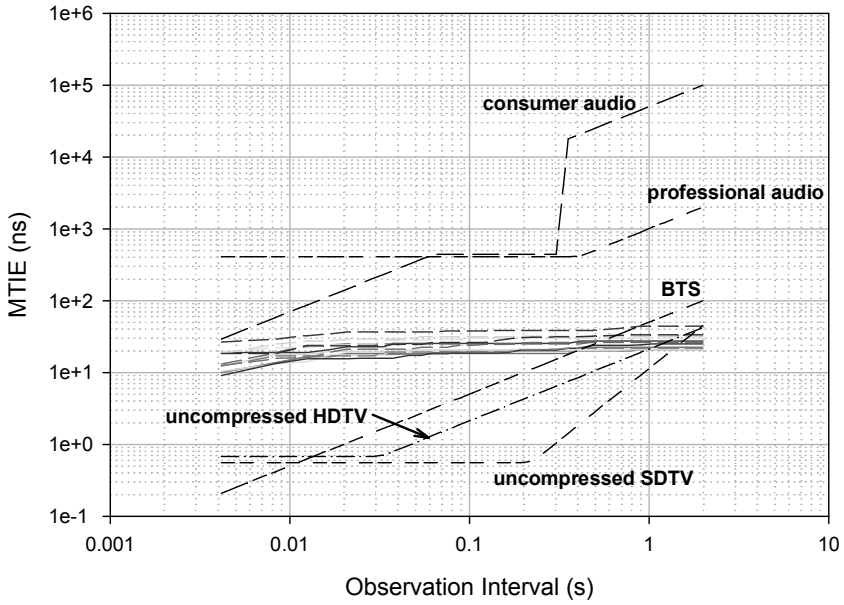


Figure 2.26.: The maximum time interval error (MTIE) for different observation intervals are shown by the bunch of dashed lines in the middle. The upper and lower dashed lines show the jitter requirements for different applications like professional video or audio. It is shown that the measured MTIEs fulfill the audio requirements [GGT09].

close to their target frequency at room temperature, but will slow down when the temperature either increases or decreases from room temperature. The crystal used for our tests was an AT-cut 25 MHz with ± 50 ppm tolerance at room temperature. A suggestion of IEEE 802.1AS standard is to use a crystal which in the worst case should not exceed ± 100 ppm. For a temperature range of 0 °C (+32 F) to +70 °C (+158 F), the crystal used is still able to maintain a tolerance of ± 100 ppm. For all AT-cut crystals, the frequency shift due to the variation of temperature can be expressed in a cubic curve. So it is expected that this crystal will work properly over a large temperature range compared to other types of crystal.

As defined by the synchronization protocol, there can be only one single best clock in an AVB network which is called the *Clock Master* (CM). The *Slave Clocks* (SCs) of the synchronized devices adjust their local clock to this clock. External climate influences like temperature manipulate each free running crystal, so they behave in a different manner. The system clock that is provided by the Ethernet transceiver may generate an additional error because the clock and data recovery circuit in each

transceiver has a jitter in the order of a few nanoseconds. Therefore, it is necessary that the offset calculation of the IEEE 802.1AS algorithm is performed periodically in order to prevent drifting clocks. A drift adjustment is done by slightly increasing or decreasing the output frequency of the crystal. Once the drift rate has been measured and compensated correctly, the slave clock offset should remain fairly constant at each synchronization interval. Finally, the slave clock uses the drift adjustment and offset correction to generate the synchronized output clock signal.

Sample Clock Distribution Figure 2.27 shows the sample clock distribution of a transceiver node, the *talker*, to the receiving node, the *listener*. Streaming data like video or audio is transported with the sample clock, e.g. 48 kHz, which is derived from the system clock. The talker of a stream uses directly the digital data or digitalizes the analogue audio data with this particular sampling rate. The digital audio data is packed together with the presentation timestamp into audio packets and is sent via the IEEE 1722 transport protocol to one or more listeners. The listener needs to regenerate the analogue audio data from the digital samples. Thus, the listener needs to recover the original audio clock. The header of the IEEE 1722 protocol embeds all necessary information for recreating the sample clock. The Listener recovers the sample clock with the use of its IEEE 802.1AS clock time, the presentation time of the audio samples, and the number of audio samples. This clock is used to drive the digital analogue converter and output the audio signal. The time calculation will be affected negatively when the crystal is oscillating in a different behavior than the fundamental. So, the difference between the presentation timestamp of two following IEEE 1722 packets or the global time could be too large and QoS cannot be provided anymore. In this case, the current implementation will cancel the stream and restart the synchronization process.

Sample Clock Jitter Jitter is defined as the variation in the delay of received packets. Ideally, packets should be delivered in a perfectly periodic-fashion. However, even if the source generates an evenly-spaced stream, unavoidable jitter is introduced by the network due to varying queuing and propagation delays. Jitter is an important QoS parameter for high-quality media streaming over a network. The following measurements will evaluate the behavior of the sample clock jitter under several temperature conditions. Since the presentation timestamps have a resolution of one

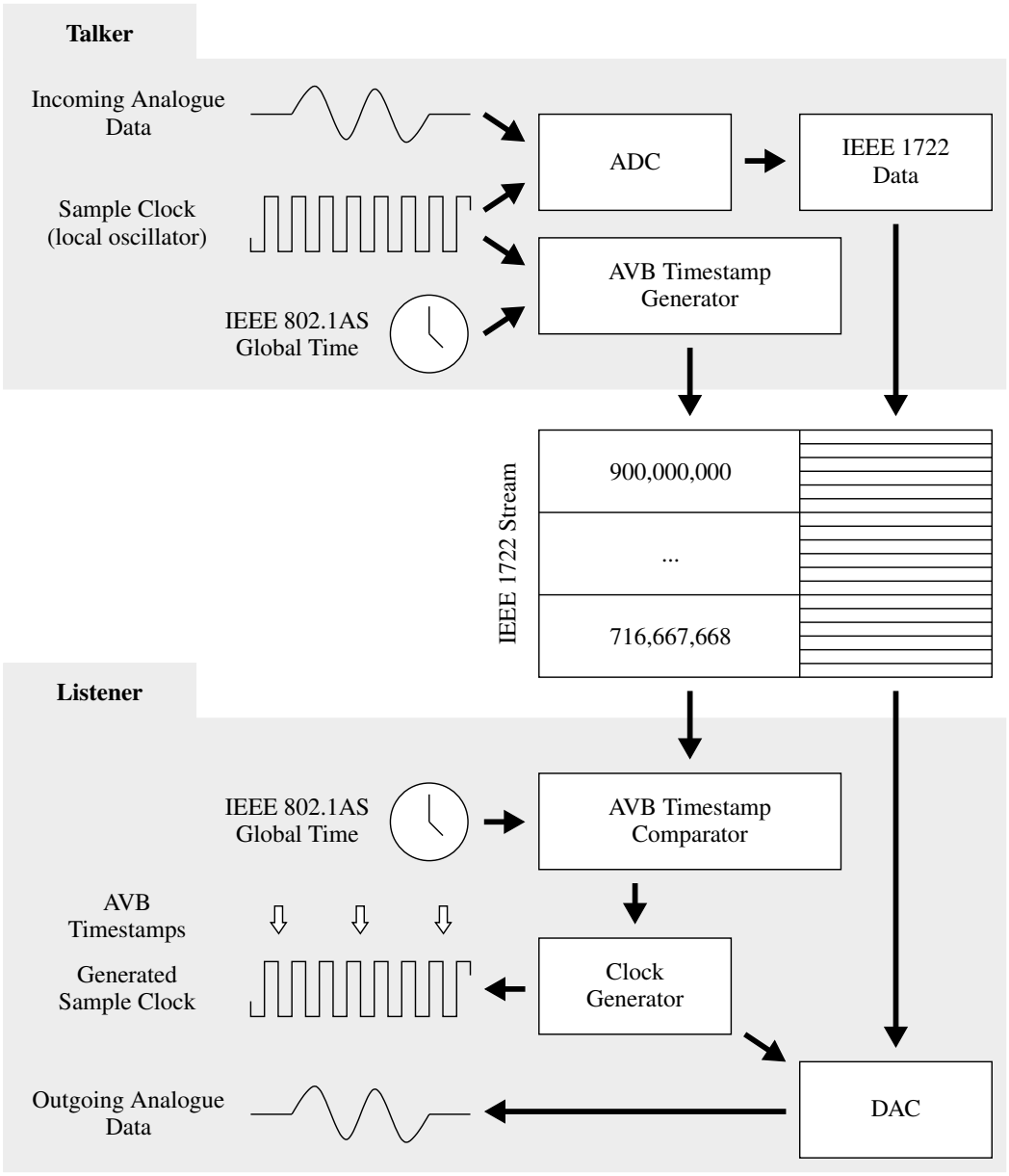


Figure 2.27.: An overview of the sample clock generation and recovery process.

nanosecond, this process introduces at least one nanosecond of jitter. It is important to realize that implementations that utilize timers running at clock rates less than 1 GHz will introduce jitter greater than one nanosecond. So, the 125 MHz core of the micro controller used by the evaluation board's clock itself introduces a jitter of 8 ns which has to be considered during the interpretation of the test results.

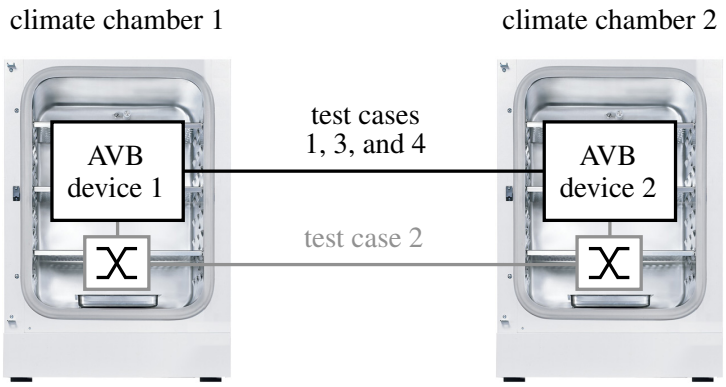


Figure 2.28.: The considered test setup for the test cases 1, 2, 3, and 4.

2.3.4. Test Cases

During the time of the measurements, the AVB substandard was still in the process of the sponsor ballot so that evaluation boards were available. For the direct connection tests, two evaluation boards from Broadcom named *AVB device 1* and *AVB device 2* are used as endpoints which were placed in different climate chambers as shown in Figure 2.28. Additionally, two switches with a Broadcom core were used to set up a full AVB network for the network test cases. The network transmission rate is 1 Gbit/s.

The goal of the tests is to measure the quality of the sample clock distribution and the number of stream distortions. For that purpose, it is necessary to set up some audio streams which are transported via Ethernet from one device to the other. Therefore, three audio streams with 48 kHz audio signals are streamed in both directions in order to draw a conclusion which part of the AVB network causes more influence of the jitter. Audio stream α denotes a 48 kHz data flow from AVB device 1 to AVB device 2. Audio stream β denotes a 48 kHz data flow from AVB device 2 to AVB device 1. Audio stream γ denotes a continuous 48 kHz data flow from AVB device 1 to AVB device 2. This stream is used for the endurance test. Table 2.2 summarizes the set up streams for the test cases.

The digitalization of analogue audio data as well as the reconversion is done by an *Analogue Digital Converter* (ADC) and a *Digital Analogue Converter* (DAC). These converters use the sample clock for the transformation. The measurement probes for the tests were positioned at the ADC/DAC sample clock line of the talker and the listener. Several tests are performed to cover a wide range of application

scenarios for in-vehicle networks. The temperature for the devices under test ranges from −10 °C (+14 F) to +70 °C (+158 F) and is measured with a temperature sensor close to the devices. This range is selected because the test devices is specified for commercial use and did not support the typical automotive temperature range from −40 °C (−40 F) to +85 °C (+185 F). All tests are done with a synchronization time interval t_s of 10 ms.

Test Case 1: Static and Slowly Varying Temperatures The first test case describes a scenario where both devices were placed in a climate chamber respectively and started at a temperature of −10 °C (+14 F). After the temperature of the first climate chamber reached the maximum temperature of +70 °C (+158 F), the temperature of the second chamber with the second device was increased in a similar manner. The temperature was changed very slowly in steps of 10 K with a maximum speed of 2 Kelvin per minute. The audio streams α and β were started after the boards have reached their respective temperature changes. Then, the time until the listener clock regenerates the sample clock was measured. From that point on, the jitter between the talker and listener sample clock was recorded for 30 seconds at constant temperature. This procedure was repeated five times for the streams α and β and each temperature value. The third stream γ was started once and recorded the jitter over the whole life cycle of the test, also during the temperature changes.

Test Case 2: Static and Slowly Varying Temperatures with Switches The second test case is similar to the first test case. No measurement was done during the temperature change. In addition to the AVB end nodes, two AVB capable switches were placed between the end nodes and inside the climate chamber to increase the number of hops. The maximum temperature had to be reduced to +40 °C (+104 F) for this test due to thermal problems of the consumer grade switches.

Table 2.2.: The specified audio streams of the test cases.

stream	source	destination	type
α	device 1	device 2	48 kHz Audio
β	device 2	device 1	48 kHz Audio
γ	device 1	device 2	48 kHz Audio

Test Case 3: Endurance Test The endurance test shows the long term stability and the long term jitter. The audio stream γ was started at the beginning and long term jitter was measured during the temperature change. This differs from the first two tests. The whole test duration was about four hours. The temperature of one climate chamber was decreased to $-10\,°C$ ($+14\,F$) while the other chamber increased the temperature simultaneously to $+70\,°C$ ($+158\,F$). These changes have been repeated several times during the measurement.

Test Case 4: Shock Test ECUs which are mounted close to the shield metal or the engine must be capable of withstanding higher temperatures than all other ECUs. Rapid temperature changes in the environment caused by snow, rain or engine start will influence those control units heavily. Car manufacturers use a special shock test to simulate such or even harder conditions. Note that ECUs today have to be operable before and after a temperature shock, but not during this shock. So, this test goes beyond the required worst case. The ambient temperature was rapidly decreased from $+70\,°C$ ($+158\,F$) to $-10\,°C$ ($+14\,F$) and vice versa in just five seconds. The audio streams α and β were started at the beginning of the test. With the start of the variation of the temperature the sample clock jitter of both streams was measured every 30 seconds for a time period of 6 minutes. In addition, those tests were repeated with a modified IEEE 802.1AS sync time interval t_s of 100 ms.

2.3.5. Experimental Results

Figure 2.29 shows the general approach of jitter measurement. Two high-speed sampling oscilloscopes are used with special heat-resistant cables to measure the sample clocks of a stream on both devices. The first channel is triggered on the rising edge of sample clock 1 on the first device. The second channel measures the jitter of sample clock 2 by setting the display mode of the oscilloscope to persistence for a given time interval t_m. The measured jitter is the interval spanned by the leftmost rising edge and the rightmost rising edge.

Test Case 1: Static and Slowly Varying Temperatures The results of test case 1 are summarized in Figure 2.30. Figure 2.30(a) shows the maximum, average, and standard deviation of the measured jitters of the streams α and β for differ-

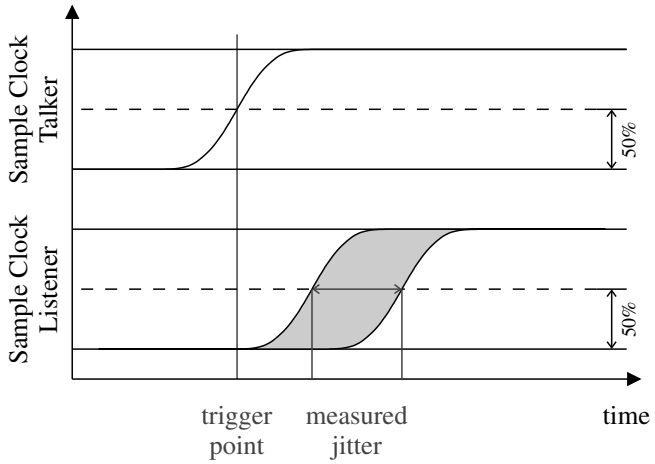


Figure 2.29.: An idealized view on the process of the clock jitter measurement. The upper channel shows the trigger point of the first sample clock 1. The second channel highlights the measured jitter interval of sample clock 2 for a time period t_m with a gray background.

ent constant temperatures. The continuous measurement of the maximum jitter of stream γ which also contained the temperature changes of the conditioning chambers are presented in Figure 2.30(b). Test case 1 points out that the maximum jitter is restricted without having any stream cancellations. In the case of constant temperatures, the maximum measured jitter was at most 18 ns and during the temperature change the measured values did not exceed a maximum jitter of 27 ns. In summary, it is shown that the synchronization of two AVB-capable end nodes operates under different temperature conditions varying from $-10\,°C$ ($+14\,F$) to $+70\,°C$ ($+158\,F$) with a restricted jitter of maximal 27 ns and no stream cancellations occurred.

Test Case 2: Static and Slowly Varying Temperatures with Switches Table 2.3 presents the measured results for test case 2. The measurement series shows that the inclusion of two switches between the two end nodes has no visible impact on the jitter values. The average measured jitters are similar to the values obtained from test case 1 without switches and only the maximum measured jitter was slightly higher at 25 ns compared to the values from the first test case. In summary, the insertion of switches had no visible effect in our measurements, the measured jitter values were also restricted to a maximum of 25 ns for temperatures ranging from $-10\,°C$ ($+14\,F$) to $+40\,°C$ ($+104\,F$), and there were no stream cancellations.

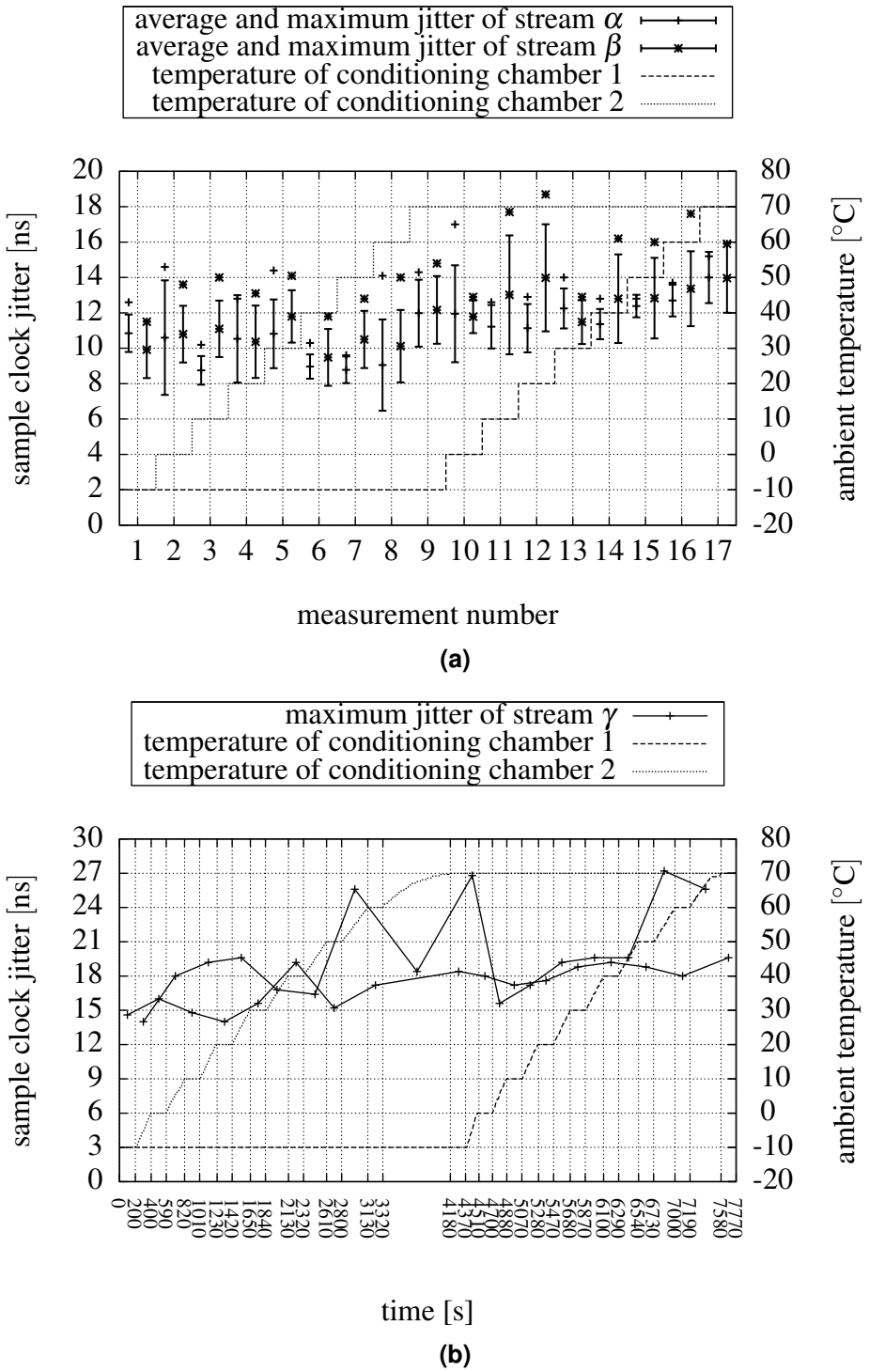


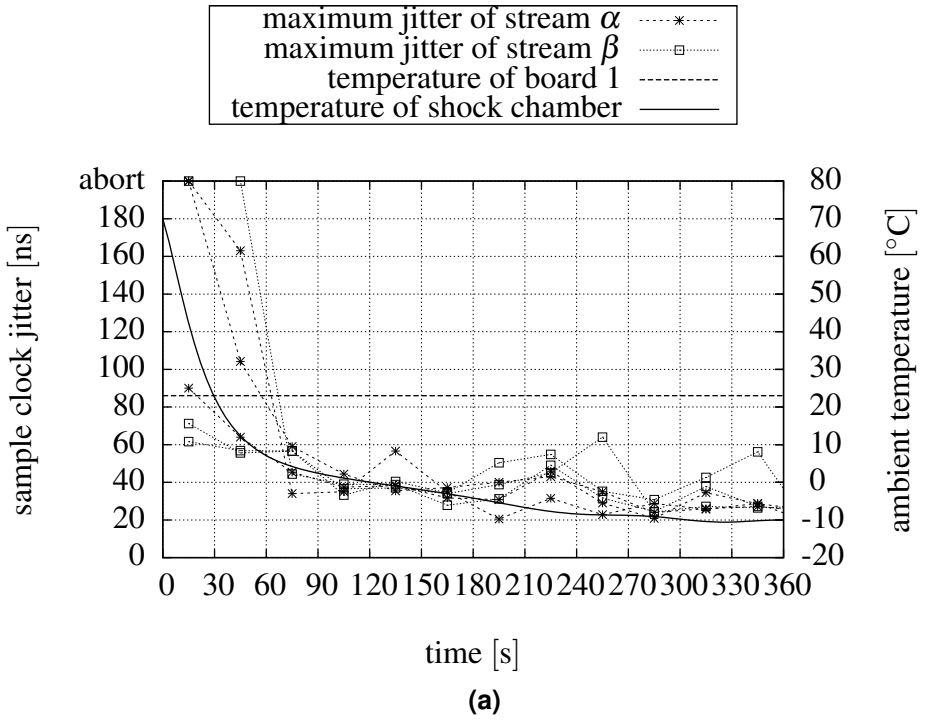
Figure 2.30.: The measured jitter values of the streams α, β, and γ for test case 1.

Test Case 3: Endurance Test In this third test case, we measured a maximum jitter of at most 57 ns over a long period of 3:40 hours without any stream cancellations. During the test run, we also changed the temperatures of the climate chambers. This shows that even a longer period over several hours leads to stable results with a restricted jitter and no cancellations of the stream.

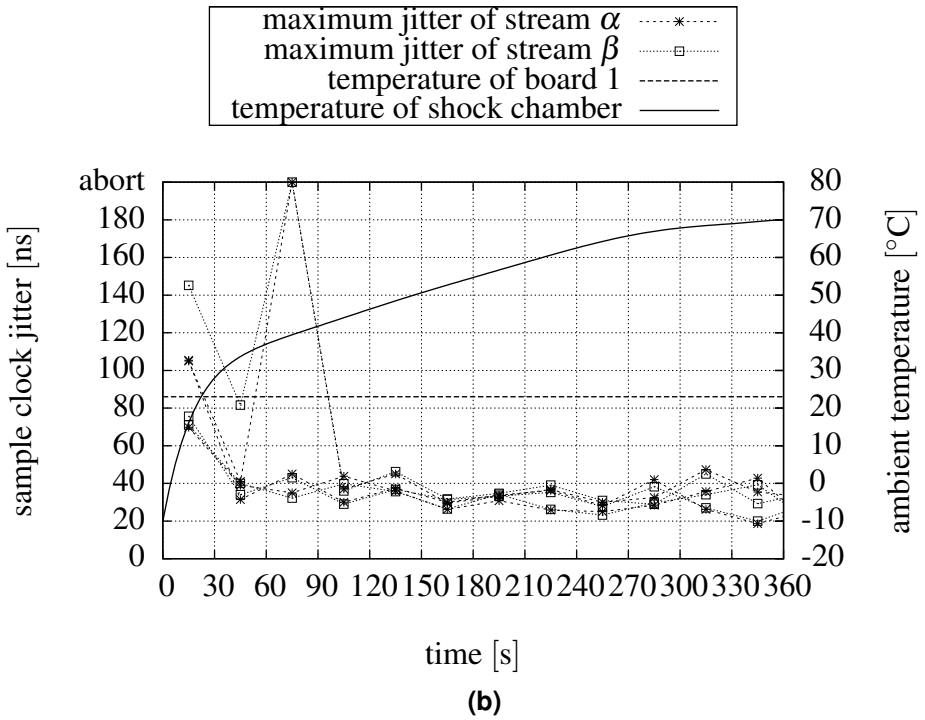
Test Case 4: Shock Test Figure 2.31 presents some results of the shock chamber measurements. The rapid temperature change from +70 °C (+158 F) to −10 °C (+14 F) is shown in Figure 2.31(a). The opposite temperature change is shown in Figure 2.31(b). Both figures point out that there is a correlation between the speed of the temperature change and the synchronization accuracy. In some test runs stream cancellations appeared during fast temperature changes. Such cancellations occur if a certain threshold in the clock drift is exceeded. In summary, this fourth test case shows that there is a correlation between the magnitude of the jitter and the magnitude of the temperature change over time. It also shows that in some cases, the synchronization algorithm was not able to maintain the synchronization between the two end nodes which is shown in Figure 2.31 by the "cancellation" entry on the y-axis. We did the same test case with a sync time interval t_s of 100 ms and obtained slightly higher jitter values compared to the 10 ms setting.

Table 2.3.: The measured jitter values of the streams α, and β for test case 2.

conditioned ambient temperature [$^\circ C$]						
chamber 1	−10	0	10	20	30	40
chamber 2	40					
jitter of stream α [ns]						
average	13.2	14.6	12.7	10.8	9.6	10.6
deviation	2.7	3.0	4.3	2.4	1.4	1.6
maximum	17.0	18.3	20.0	14.4	11.6	13.1
jitter of stream β [ns]						
average	13.9	16.9	12.5	11.2	11.5	9.3
deviation	1.5	4.0	2.6	1.4	2.5	1.5
maximum	16.6	24.6	15.6	13.1	15.6	12.0



(a)



(b)

Figure 2.31.: The measured jitter values of the streams α, and β with a synchronization time of 10 ms for test case 4.

2.4. Throughput Performance Aspects of Resource-limited Systems

New emerging applications like video- or audio-based assistance functions will require higher network throughputs in the future. In order to enable throughputs of up to 100 Mbit/s and above on resource-limited embedded systems, special hardware and software enhancements are necessary. In the following, different concepts are presented, implemented, and evaluated to show that even throughputs near line-rate of Gigabit Ethernet can be achieved. After introducing the concepts and the test environment, the measured throughputs are presented and discussed.

2.4.1. Fundamentals

Todays automotive communication technologies LIN, CAN, and FlexRay have a limited physical data rate of up to 10 Mbit/s. This relatively low data rate has the advantage that resource-constrained processors are able to provide these throughputs up to line rate. In the case of Ethernet, the minimum provided physical data rate for automotive use cases is 100 Mbit/s and several resource demanding protocols like IP, UDP, and TCP have to be supported. Comprehensive measurements show that the computing power of typical embedded processors is not able to provide these high data rates. The following evaluation results are based on an evaluation platform consisting of two evaluation boards with an embedded processor running at 85 MHz. The two boards are connected via an Ethernet connection. The throughput, latency, and jitter measurements were done using 10, 100, and 1000 Mbit/s Ethernet. The complete test setup, measurement concept as well as the evaluation results are summarized in [Ker08•].

Throughput In communication systems, throughput is a measure of available and consumed communication resources in bits per second. The physical throughput of a communication technology like 100 Mbit/s in the case of Fast Ethernet denotes the number of bits that can be transmitted per second. In general, it does not include the frame structure or any additional network protocols. The available application throughput – the throughput that remains for application-specific information – denoted with b_{raw} and measured in Mbit/s can be calculated by subtracting the pro-

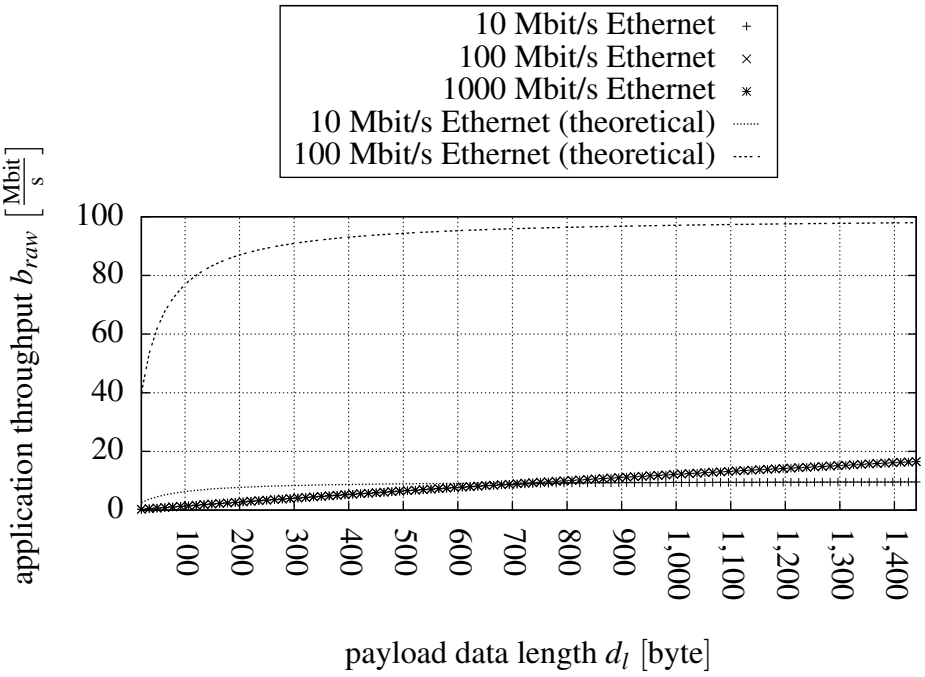


Figure 2.32.: The maximum measured application throughput for different Ethernet speeds and application payload lengths. In addition, the maximum theoretical application throughputs for 10 Mbit/s and 100 Mbit/s Ethernet are shown by the dotted lines.

tocol overheads per packet. This implies that the selected packet size has a significant influence on the available throughput. Figure 2.32 shows the measured application throughputs when using Ethernet, IP, and UDP to transport application specific data. The throughputs are measured for different Ethernet speeds and packet sizes. It shows, that the maximum achievable application throughput is about 18 Mbit/s and hence far below the 100 Mbit/s or 1000 Mbit/s for Fast or Gigabit Ethernet. Only in the case of 10 Mbit/s Ethernet, the full line rate can be reached. The two dotted lines show the theoretical maximum application throughputs for 10 and 100 Mbit/s Ethernet.

Latency The end-to-end network latency of a distributed application system is a measure of time delay in seconds. It includes the time duration of the network stack at the transmitter and receiver as well as the delay within the network. Figure 2.33 shows the layered end-to-end network latency based on the OSI-model for different

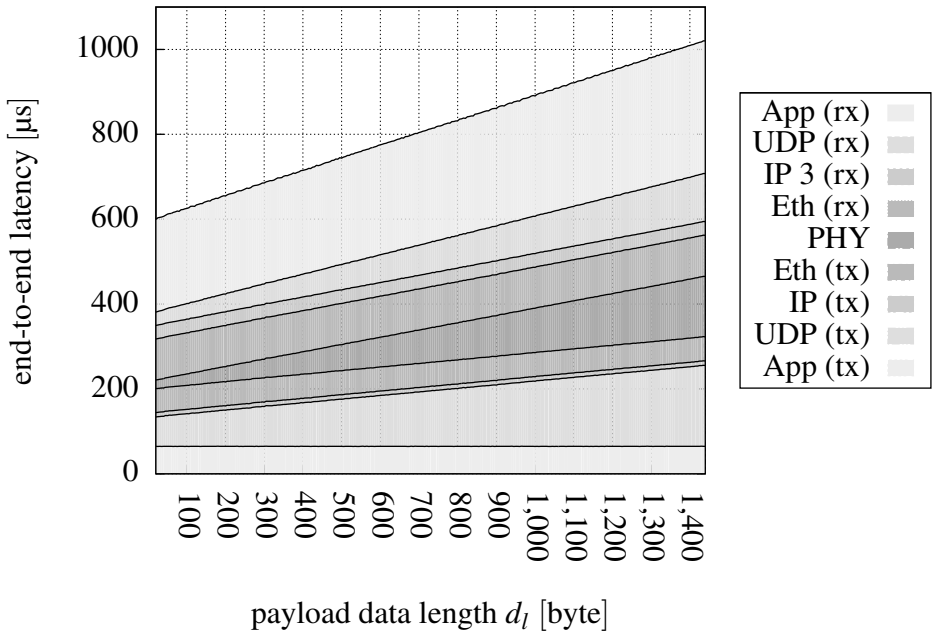


Figure 2.33.: The end-to-end latency of packets going from one application to another for different application payload length. The intervals are based on the OSI model layers and show the time duration within each OSI layer and on the physical media.

packet sizes d_l. The latency is proportional to the packet size and most of the time is spent in the software network stacks on transmitter and receiver side. The processing times present the throughput bottleneck. The processor needs to much processing power and time to prepare all the protocol headers before sending the packet. Moreover, the calculation of the transport protocol header and payload checksum requires a lot of time compared to the other protocol blocks.

Jitter In telecommunication systems, jitter is the undesired deviation of an assumed periodic signal or message. The periodic jitter J_p of a message m with an assumed cycle time $T_0(m)$ is calculated by Equation (2.10). T_{per} denotes the measured cycle time of the actual cycle n.

$$J_p(m) = T_{per}(m,n) - T_0(m) \tag{2.10}$$

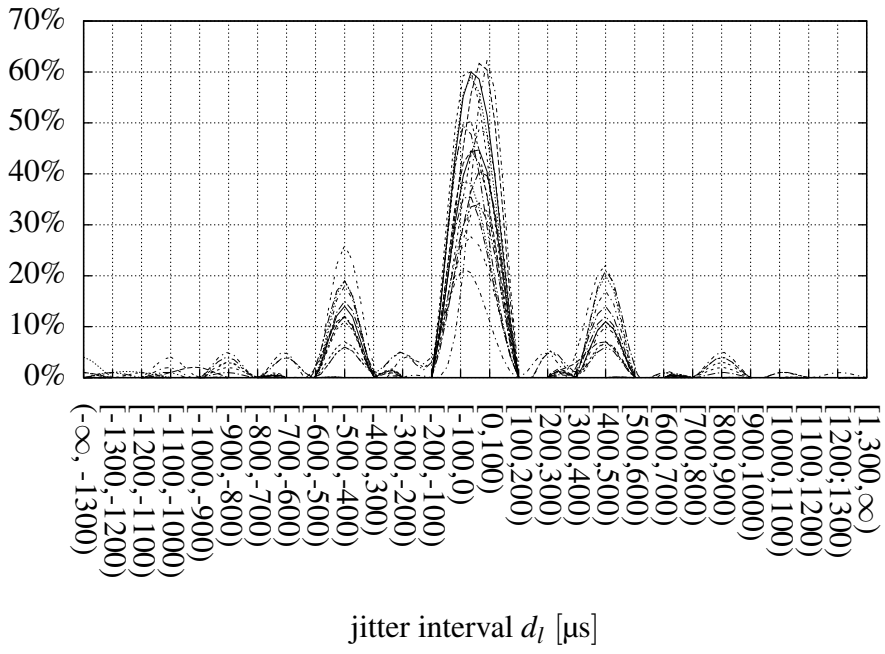


Figure 2.34.: The measured jitter J_p distribution for seventeen different cyclic messages send by one node.

The maximum periodic jitter J_p^{max} of a message m is defined by Equation (2.11):

$$J_p^{max}(m) = \max_{\forall n} |T_{per}(m,n) - T_0| \tag{2.11}$$

Figure 2.34 shows the measured percentage distribution of the deviation of the periodic jitter for 17 different messages. The messages were assumed to have different cycle times. It is shown, that the prototype caused jitter values for the different messages in the range of $-1400\,\mu s$ to $1400\,\mu s$. In summary, it is shown that in 60 % of all cases, the packet jitter was within a range of 0 to $100\,\mu s$.

2.4.2. Related Work

A good overview of hardware- and software-related throughput measurement and optimization topics is presented in [Alt09]. The authors discuss techniques for improving performance by using FPGA-based systems. Discussed optimizations include a faster processor core, memory with low latency and fast access times, increased data and instruction cache sizes, the choice of the optimal period for system clock

timers, and the partial increase of clock frequency. Todays fast embedded processors already support several features like instruction caches, data caches, dynamic branch prediction, hardware multiplication, hardware division, and barrel shifter to optimize their throughput. A large impact on the overall system performance is caused by memory access time and latency. The goal is to minimize the number of processor instructions per memory read or write access. Typical systems provide large low speed and small high-speed memories. Thus, critical code and data sections should be placed in high-speed memories and vice versa. Networking involves a numerous number of data copies and manipulation of the same data. Therefore, increased data and instruction caches can help to speed up these operations. The choice of an optimal system clock timer helps to minimize timer interrupts. For example, every time the system timer interrupt occurs, two cost intensive context switches have to be performed. If the period is shorter than it needs to be, an unnecessary percentage of cycles is used for handling these interrupts. Finally, a partial increase of the clock frequency can have a significant influence on the overall performance. For example, by isolating the data path from the FPGA to the Ethernet MAC/PHY, a higher throughput can be achieved. In addition, they present some simple performance results for different implementations. The presented maximum transmit throughput of a TCP connection is 63 Mbit/s for packets with a payload size of 1024 bytes. In [Alt10], the authors present an application node based on the optimization techniques from above for implementing an Ethernet optimized design example. The resulting maximum transmit throughput is 81 Mbit/s for an application running on an 85 MHz processor. Additional techniques like interrupt coalescing, checksum offloading, zero-copy data movement, jumbo frames, and DMA are presented in [CGY01]. *Interrupt coalescing* is a method to selectively delay interrupts if more packets are pending for transmission for example. This helps to reduce the overall number of interrupts which have to be processed by the CPU. The offloading of checksum calculation from software to hardware also accelerates the processing of the network stack. In special, the header and payload checksums of transport protocols lead to high latency due to the huge amount of data which has to be considered when calculating the checksum. Zero-copy data movements reduce the number of data movements in memory. Most problematic is the transition from user memory space to operating system memory space. They discuss additional techniques like *page remapping* or *page flipping* to solve this problem. Jumbo frames can be used to improve the payload ratio of a

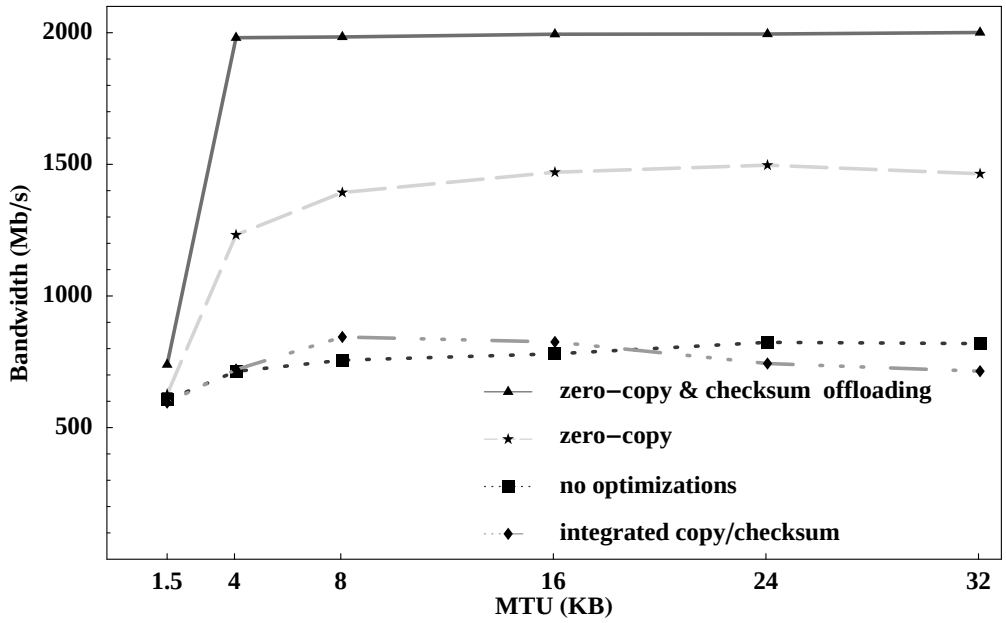


Figure 2.35.: The maximum achieved throughput for different optimization stages and payloads measured on PowerEdge 4400 platform. The dotted line represents the reference implementation. The other lines show the throughput improvements for the integrated copy/checksum, the zero-copy, and the zero-copy with checksum offloading approaches [CGY01].

packet to minimize per packet overheads. Finally, intelligent DMA mechanisms are introduced to release the processor from just copying data from one memory space to another. Figure 2.35 summarizes the results measured on a PowerEdge 4400 platform and shows the average measured throughputs from 50 test runs for different Ethernet frame sizes. For an Ethernet *Maximum Transmission Unit* (MTU) of 1500 bytes, a maximum throughput of about 90 Mbit/s is presented. A FPGA-based concept for interfacing a 100 Mbit/s Ethernet link is discussed in [Ins04]. The author discusses most of the techniques mentioned above and introduces the use of a multiplexer allowing high data rate traffic beside the conventional method by directly accessing the Ethernet MAC from the processor. This enables separate hardware modules to send and receive Ethernet traffic without interrupting the main processor. Beside the concept, no experimental results are presented.

In summary, it is shown that the networking throughput of low cost embedded systems is limited by the software processing overhead per sent packet and cannot be significantly raised through CPU and software optimizations. Satisfactory networking speeds are only achievable by using fast general purpose processors or dedicated hardware blocks.

2.4.3. Throughput Tuning: System Variants at Transmitting Side

All of the following system variants are based on a standard Ethernet reference design from Altera [Alt10]. To assess the impact of the tested system modifications, the performance throughput of each variant is compared to the reference design. Table 2.4 gives an overview of the considered system variants. Variant 0 is the reference design, variants 1a, 1b, and 1c make use of software modifications, variant 2 uses standard hardware adaptions, and variant 3 is based on an architecture with special hardware blocks to show the maximum achievable throughput.

The main task of the system is to process data from a streaming *data source* and to transmit the data over the Ethernet MAC interface. For example, the data source can be a video or audio source.

Variant 0: Reference Design The configuration of the reference design is shown in Figure 2.36(a). In this case, the data source is directly connected to the *Double Buffer Direct Memory Access* (DBDMA) module which writes the incoming data

Table 2.4.: The examined system variants.

Variant	Description
0	reference design
1a	special cache strategy
1b	disabled UDP checksum
1c	disabled UDP checksum and expanded caches
2	packet generation in hardware
3	packet generation and transmission in hardware

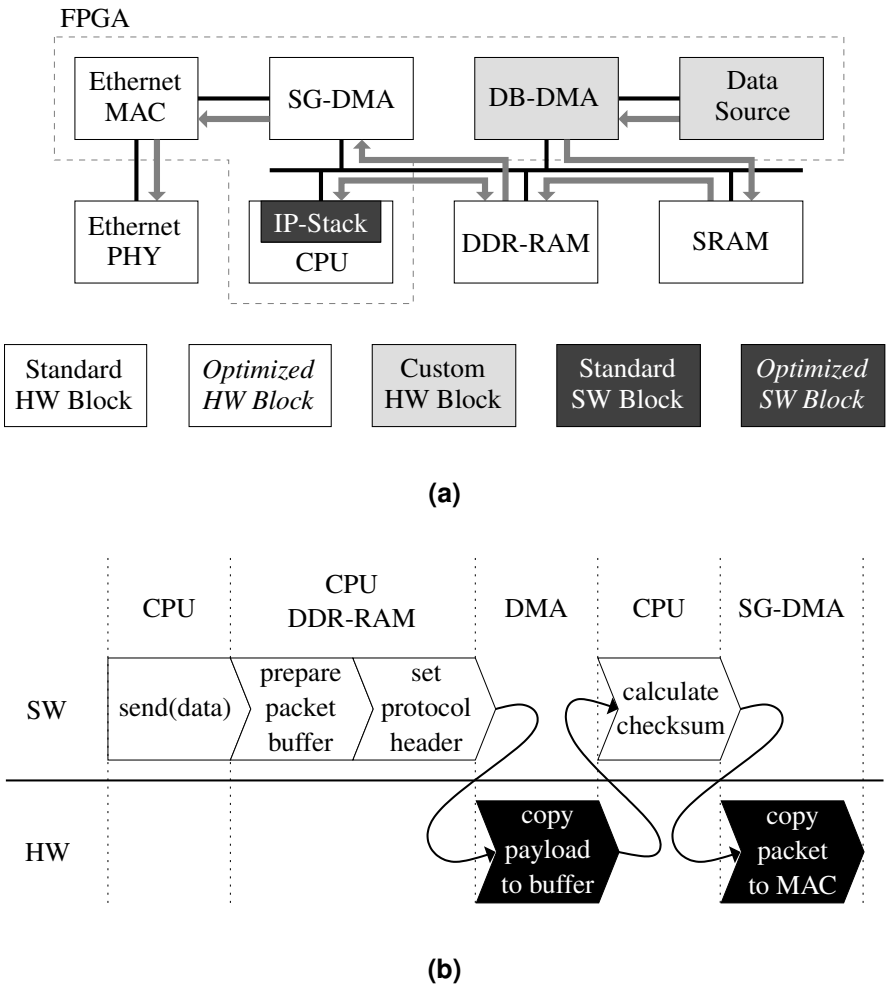


Figure 2.36.: Reference Design: (a) Shows the architecture of the reference design and (b) presents the process chain for a standard packet transmission.

into the *Static Random Access Memory* (SRAM). The data source streams application data to the DBDMA which alternately writes the data into one of the two buffers. If one buffer is full, the DBDMA locks the full one, notifies the software running on the CPU via an interrupt which then starts processing the payload by sending out UDP packets on the Ethernet MAC interface. Figure 2.36(b) shows the resulting event chain. In the first step, the network stack prepares a packet buffer and configures the headers of the different protocol layers. The packet structure is located in the general system memory which is the *Double Data Rate Random Access Memory* (DDR RAM) in this case. Since the packet buffer and the payload are located in physically and logically separated memory areas, the payload has to be copied into the packet

structure. Here, the SRAM is used due to its short read and write delays and the copy process can be supported by a DMA to reduce CPU load. The calculation of the UDP checksum concludes the preparation of the packet. After that, the software configures the *Scatter Gather DMA* (SGDMA) which then reads the packet from the DDR RAM and writes it directly to the Ethernet MAC. An SGDMA consists of a linked list which is processed in a *first in first out* manner and contains *descriptors*. Each descriptor includes a source and destination address and the length of the data which has to be copied. The SGDMA then processes the list and executes the copy instructions defined in each descriptor. The advantage of an SGDMA is that it is not necessary to wait until the currently processed copy instruction has finished before adding the next one.

Variants 1a, 1b, and 1c: Caching and Software Enhancements In the following, the variants 1a, 1b, and 1c are described. The design changes affect the configuration of the CPU and caches as well as the software network stack. Figure 2.37 shows the underlying architectures and the adjusted process chains for the send call of the different variants. Variants 1b and 1c omit the UDP checksum calculation.

Variant 1a uses an optimized caching strategy for the instruction and data memory. The achievable network throughput is mainly limited by the performance of the TCP/IP software stack. To reduce the latency caused by the traversal of the software stack for each packet sent, the processing speed of the code has to be increased. The performance and architecture of the system can be tailored towards the system requirements at the expense of additional resource usage. One possibility of increasing performance while keeping the clock rate constant is the addition of *tightly coupled instruction memories* (TCIMs). TCIMs are memories that are directly attached to the CPU and cannot be accessed by any other component. Since TCIMs are implemented as on-chip memories, access latency and speed are comparable to those of an instruction cache. Software fragments can be moved to the TCIM through source code annotations that are recognized by the compiler. A code that resides in a TCIM cannot be cached and is always loaded from the TCIM. This frees up the regular instruction cache and more importantly guarantees a constant execution and load time. For example, this removes the uncertainty of the execution time of software network stack. In our case, one main task of the application software is the recognition and the processing of a full data buffer. To evaluate the benefit of adding a tightly coupled

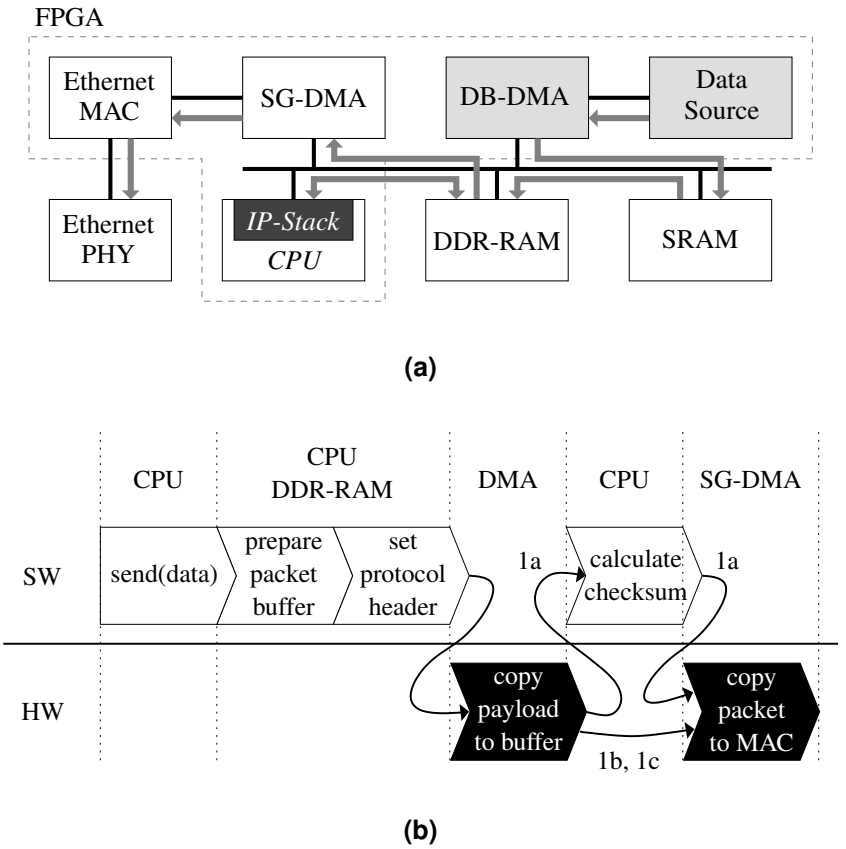


Figure 2.37.: Variants 1a, 1b, and 1c: (a) Shows the architecture and (b) presents the process chain representing the transmitting node latency for packet transmission.

instruction memory, the buffer handling code is relocated to the TCIM. No additional hardware changes have to be made, since an unused TCIM is already included in the reference design.

Variant 1b additionally disables the UDP checksum calculation. When sending data over a standard UDP socket, the network stack calculates a CRC checksum over the entire UDP packet including the header and payload. This is necessary to detect and discard erroneous packets at the receiver. An automatic retransmission of damaged frames is not supported by the UDP protocol. Any further error handling has to be done by the user application. The calculation of the UDP checksum can also be disabled through options which are set upon creation of a new socket. Damaged Ethernet frames are still recognized and discarded on the data link layer since the Ethernet frame checksum is calculated in hardware and is therefore still activated.

With regard to a streaming application, an erroneous packet will generally not affect the functionality of the application so the UDP checksum could be disabled. Figure 2.37(b) shows the adjusted send call with omitted UDP checksum calculation.

Finally, variant 1c uses expanded caches. Since the software structure of our system is kept rather simple, even simple caching strategies should deliver good results as long as the instruction cache provides sufficient space. To avoid the replacement of cached instructions even if they are called very frequently, the size of the instruction cache is doubled from 4 kilobytes to 8 kilobytes. Therefore, this variant does not modify the general system or software architecture, but examines the impact of a bigger instruction cache which should result in a higher cache hit and miss ratio. In turn, this should reduce the execution time of the buffer handling and the packaging of data by the software stack. Additionally, the network stack jitter should become smaller.

Variant 2: Packet Generation in Hardware Although a software network stack provides maximum flexibility, it comes at the expense of increased system requirements which cost-intensive embedded systems struggle to meet. One of the main causes of latency is the traversal of the software network stack. Considering the topology of automotive communication networks, parameters like the number of network nodes and the network topology itself are known at design time and remain static even at runtime. With regard to a streaming application which is designed for automotive applications, one can assume that the sources and the receivers are not going to change during runtime. Therefore, our system only needs to handle changing senders or receivers during the startup and configuration phase, whereas transmission of streaming data targets fixed network nodes. If the network is restricted to static IP and MAC addresses, the source and destination addresses are already known and do not have to be detected dynamically at runtime.

Furthermore, by using a fixed packet size for the transmission of camera data, omitting the CRC calculation for UDP packets, and disabling the IP fragmentation, the protocol headers of UDP, IP, and Ethernet remain static. Figure 2.38 shows the data structures of the different protocols with their specific properties regarding to our considered streaming use case.

The header can then be generated prior to the data transmission and is prepended to a given amount of payload data, thus, creating a complete packet in hardware.

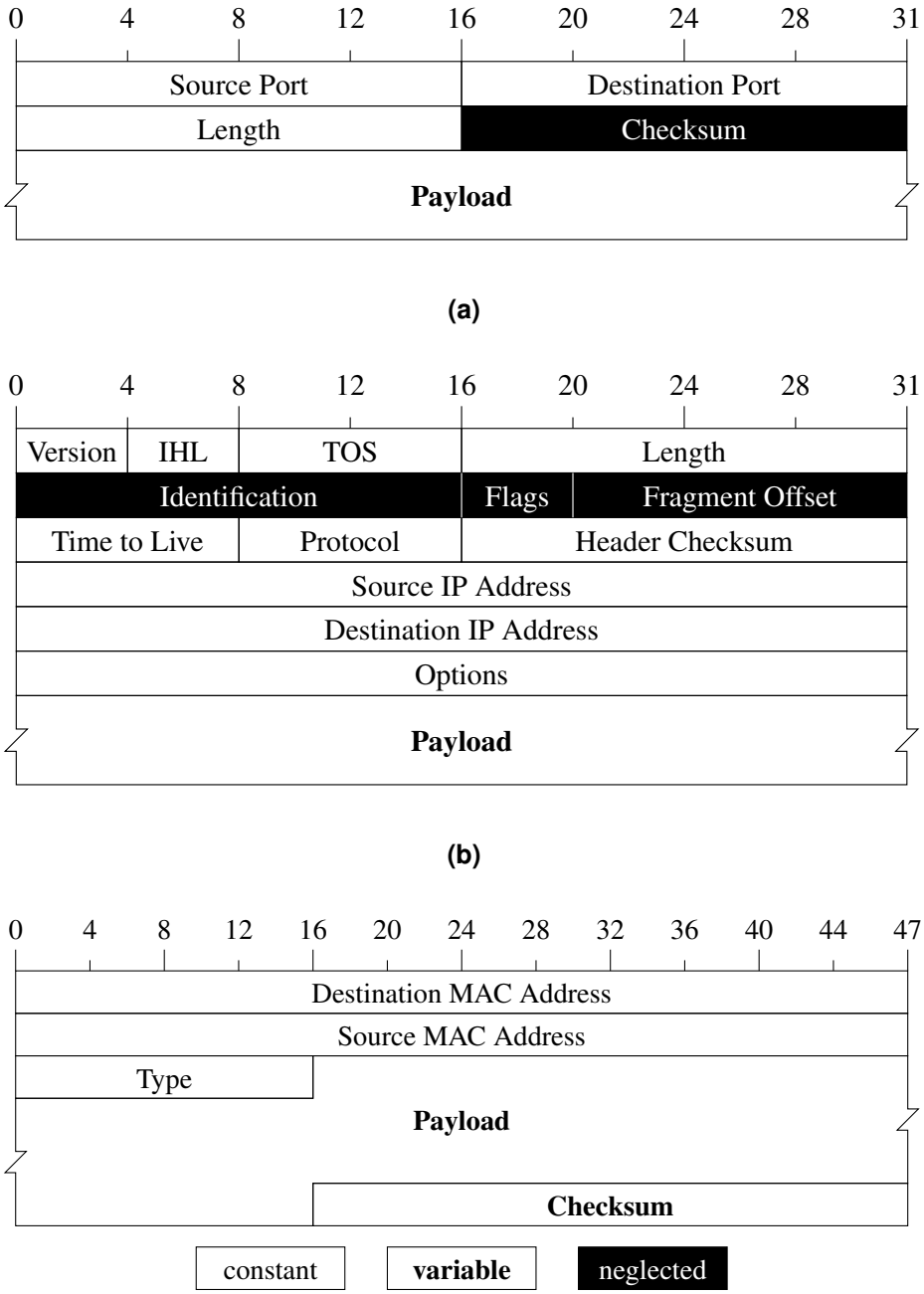


Figure 2.38.: The data structures of the protocols: Black protocol fields are neglected by the considered system. (a) Shows the UDP protocol header. (b) Shows the IP protocol header. (c) Shows the Ethernet header.

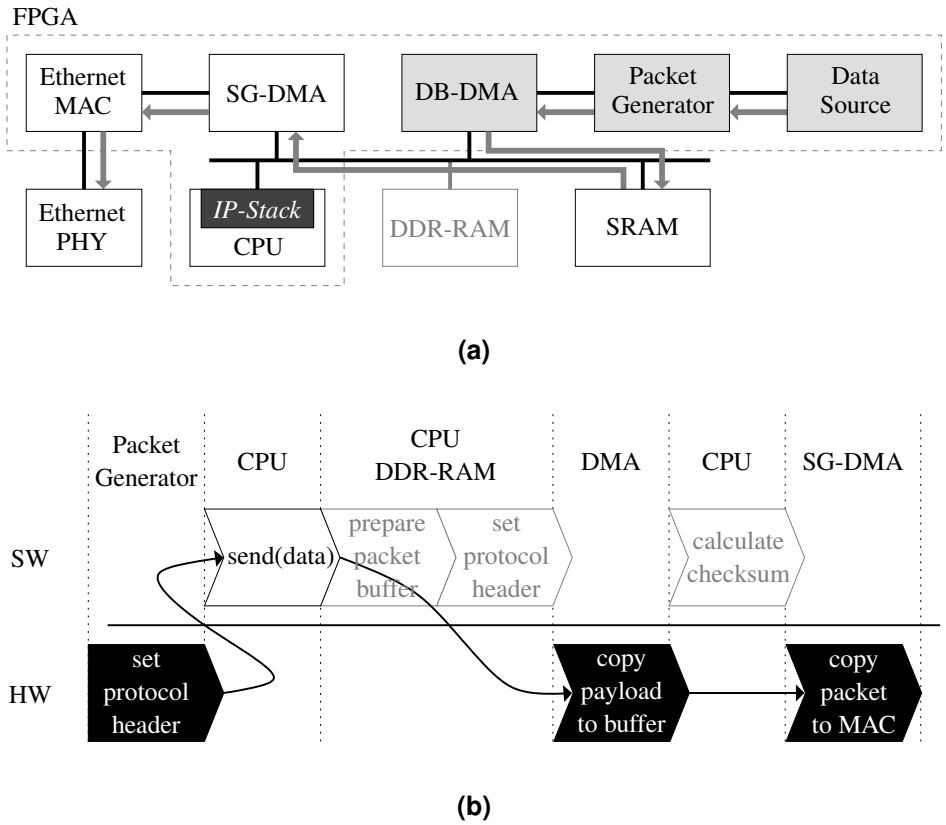


Figure 2.39.: Variant 2: (a) Shows the architecture and (b) presents the process chain for packet transmission.

Figure 2.39(a) shows the changed system architecture. The added *packet generator* contains the preconfigured header and inserts it into the data stream. The DMA then writes the complete packets into the buffer. Similar to the approach described above, the software starts processing the buffer once it is filled. However, no further intervention of the network stack is needed and the packet can be written directly to the MAC by the SGDMA. This has a considerable impact on the CPU load since the necessary interaction for the transmission of one packet is greatly reduced, as shown in Figure 2.39(b). The protocol headers are automatically inserted by the packet generator. The transmission of the raw packet is initiated by an interrupt which adds an entry to the processing queue of the SGDMA consisting of a source address, destination address, and length of the raw data. Configuring and adding the entry is the only task that has to be done in software.

Variant 3: Packet Generation and Transmission in Hardware Taking into account that a complete UDP packet is being created for the raw send function approach from variant 2, one could further eliminate the dependency on the CPU and perform packet transmissions completely in hardware, as shown in Figure 2.40(a). A multiplexer is used to share the MAC between the CPU generated Ethernet frames and the streaming application based Ethernet frames generated in hardware. Thus, the change is transparent and has no implications on the rest of the system. To ensure that a multiplexer input cannot block the other input, a round robin scheduler is used. This way, each channel can access the MAC with a latency of one packet, even if the bus is fully utilized. The transmission of the data is handled completely in hardware. This removes the need for any interaction on the software side as seen in Figure 2.40(b). Apart from the initial configuration during start up, this removes virtually all CPU load associated with data transfer and provides a major speed improvement as we will experimentally show in Section 2.4.5.

2.4.4. Test Environment

The test environment consists of a transceiver and a receiver node. The transceiver is an embedded *Nios 2* development board, which includes a *Stratix II EP2S60* FPGA, 16 MB DDR RAM, 2 MB SRAM, and a dedicated Gigabit Ethernet PHY [Alt09]. The Ethernet PHY is connected to a Gigabit Ethernet MAC that uses SGDMAs to transfer data to and from the network interface. A *System on Chip* (SoC) design is implemented on the FPGA. It uses an Altera Nios II softcore CPU which is running at a clock speed of 85 MHz and is configured initially with a 4 KB instruction and data cache. The DDR RAM is used as a general system memory and also contains the program code. As an operating system, the real-time operating system MicroC/OS-II including an *Iniche* TCP/IP network stack is used. All hardware components are connected via an Altera Avalon-Bus. Depending on the kind of bus access, it is distinguished between master, slave, and streaming interfaces. Note that a streaming interface is not connected to the Avalon-Switch-Fabric and represents a directed point-to-point connection between two components. The different discussed optimization concepts are implemented on the transmitting node and therefore generates and sends test data to the receiver. A computer represents the receiving side and processes the incoming packets and validates the test data. A command line tool is used

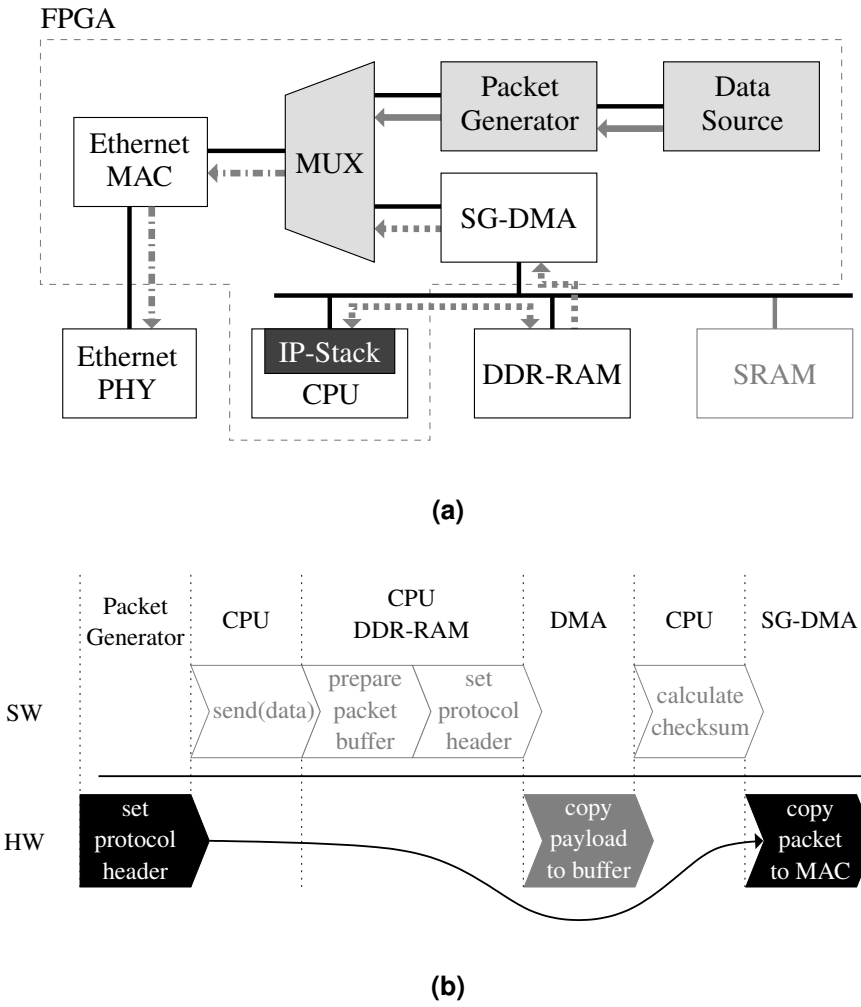


Figure 2.40.: Variant 3: (a) Shows the architecture and (b) presents the process chain for packet transmission.

to control the benchmarks. A benchmark run is defined by its packet payload size, the total amount of test data, the speed of the data source, and the chosen system variant. By focusing on streaming applications, the test system does not measure the average network throughput. Instead, it tries to find the maximum sustainable speed of a streaming data source (for example a video or audio stream). If the data source has to be stopped, the benchmark fails. Therefore, it also accounts for the influence of the bus load and the handling of the payload. The measured application data rate is equivalent to the achievable networking throughput of the system. All given throughput values are application data rates and do not include protocol overhead. Thus, the

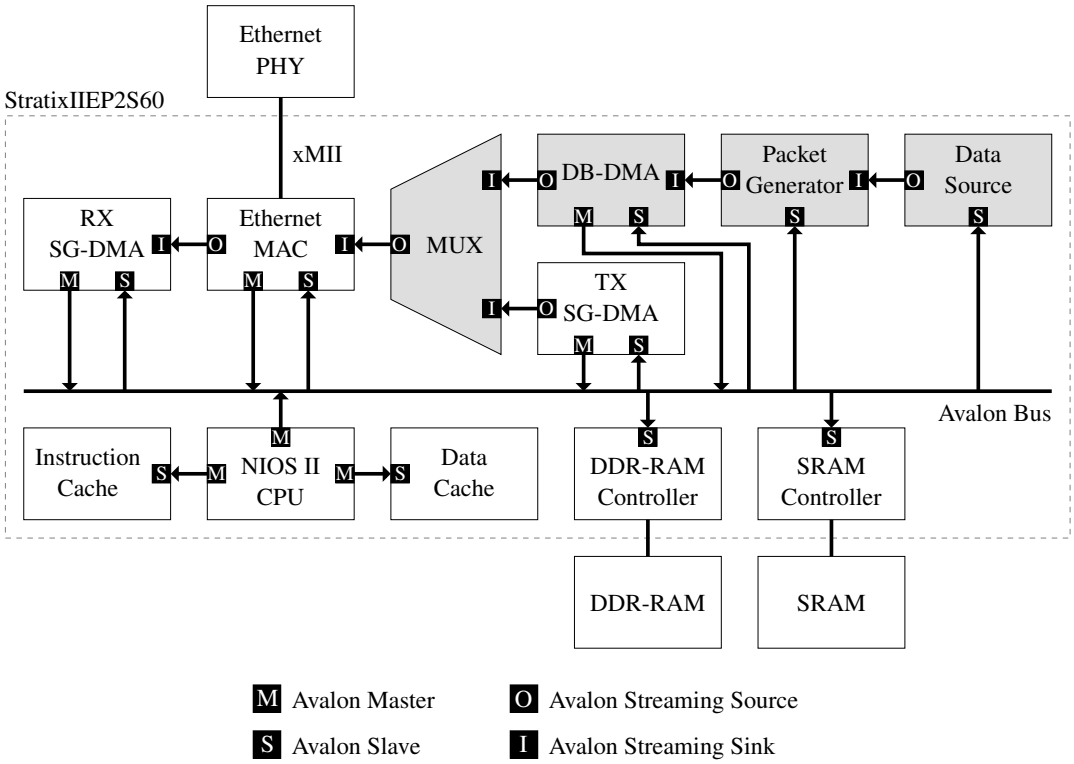


Figure 2.41.: The implementation of variant 3 on an Altera FPGA-based development board with a NIOS II CPU and an Avalon bus interconnection block.

raw data rate is higher. As an example, Figure 2.41 illustrates the implementation of variant 3 on the Altera development board. The instantiated hardware blocks and their interconnections via an Avalon-Bus and dedicated point-to-point connections are shown.

2.4.5. Experimental Results

Table 2.5 summarizes the maximal achieved throughput for different payload lengths and the proposed system variants. Sending UDP packets via the reference design results in a maximum speed of 38.31 Mbit/s at a maximum packet payload size of 1472 bytes. The throughput limitation is caused by the software network stack which needs more time for preparing one packet than sending it via the Ethernet MAC interface.

Variants 1a, 1b, and 1c: Caching and Software Enhancements Variant 1a with relocation of the buffer handling code into the TCIM caches shows just a slight improvement compared to the reference design and the results are slightly smoothed across the different packet payloads. Omitting the calculation of the UDP checksum in variant 1b increases the data rate by 16.5 Mbit/s to 54.81 Mbit/s. The enlargement of the instruction cache in variant 1c from 4 KB to 8 KB shifts the maximum throughput to 59.61 Mbit/s. Apart from omitting the UDP checksum calculation, simple software and hardware modifications do not show a noticeable throughput improvement and are not able to support a full 100 Mbit/s Ethernet connection. The results for different payload lengths of the three variants 1a, 1b, and 1c, the reference design and the maximum theoretical throughput of an Ethernet- or UDP-based transmission are summarized in Figure 2.42.

Variant 2: Packet Generation in Hardware Variant 2 shows a significant speed-up by using statically predefined protocol headers. By adding the predefined UDP, IP, and Ethernet header to the payload using dedicated hardware, most parts of the software stack can be bypassed. This leads to a significant speed-up because of a reduced software stack functionality. The maximum throughput limit is 339.98 Mbit/s for a payload size of 1472 bytes per packet which is up to 301.67 Mbit/s (8.9x) faster than the reference design. The throughput decreases almost linearly as the packet

Table 2.5.: The measured application throughputs in Mbit/s using Ethernet, IP, and UDP protocol headers for different application payload lengths in byte.

Variant	Payload Length [*Byte*]			
	18	500	1000	1472
0	0.76	18.13	30.22	38.31
1a	0.76	18.16	30.37	38.64
1b	0.76	21.51	39.98	54.81
1c	0.95	23.77	43.63	59.61
2	1.76	118.69	236.07	339.98
3	181.68	842.21	894.48	912.31

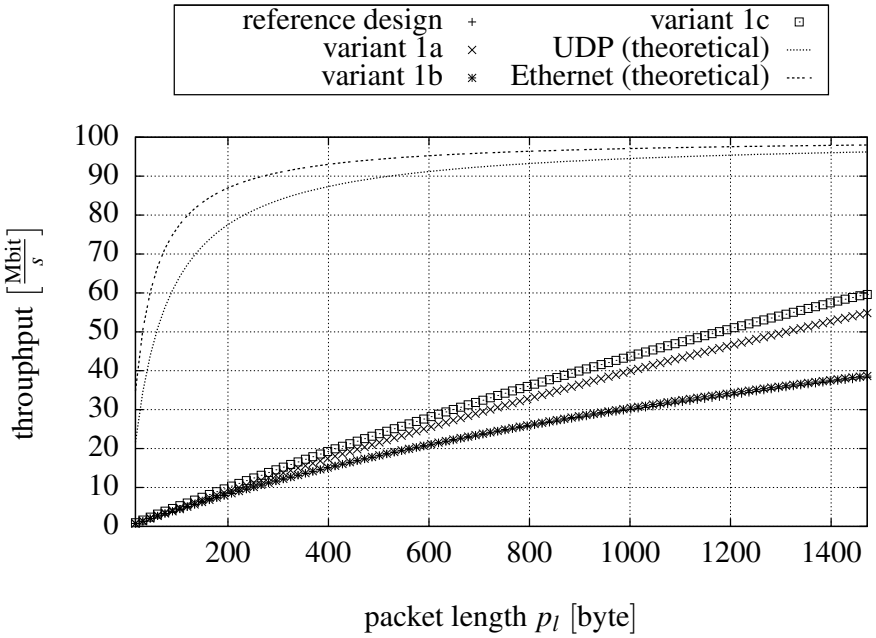


Figure 2.42.: The measured throughputs of the reference design compared to the variants 1a, 1b, and 1c.

size drops. Figure 2.43 presents the measured throughputs of variant 2 together with the reference design and the theoretically achievable throughputs.

Variant 3: Packet Generation and Transmission in Hardware By offloading the packaging and transmission of packets completely into hardware, variant 3 achieves up to 91 % of the available throughput of a Gigabit Ethernet. The user data rate achievaes a speed of up to 912.31 MBit/s. This is an increase of more than 874 MBit/s (23.8x) compared to the reference design. Figure 2.44 summarizes the measured results of variant 3 compared to the reference design as well as the theoretical maxima. The throughput of variant 3 converges to the theoretical maximum throughput of a UDP-based transmission. The raw Ethernet frame-based throughput without UDP and IP protocol headers can be calculated as follows, where p denotes the payload size of the packet and b_{udp} the measured UDP-based throughput:

$$b_{eth}(p) = \begin{cases} \frac{p+28}{p} \cdot b_{udp}(p) & 46 \leq p \leq 1472 \\ \frac{p+28}{64} \cdot b_{udp}(p) & 18 \leq p < 46 \\ b_{udp} & 0 \leq p < 18 \end{cases}$$

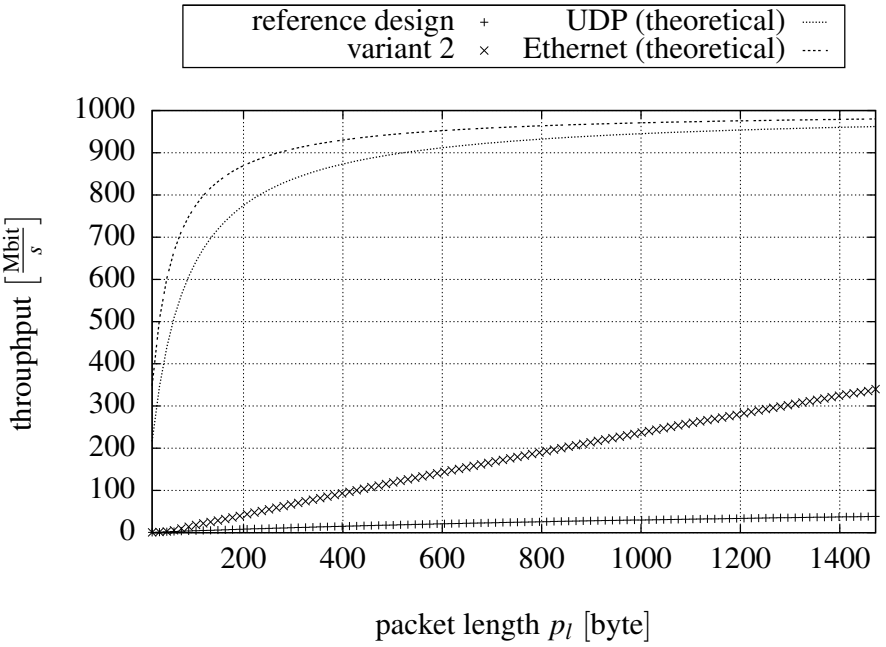


Figure 2.43.: The measured throughputs of the reference design compared to variant 2 (packet generation in hardware).

The considered system concepts generate an obvious enhancement of the maximum achievable throughput at the expense of upcoming boundary conditions. For example, when checksums are neglected, the error detection capabilities decrease. Statically predefined protocol headers constrain the system design capabilities, when, for example, addresses or packet sizes have to be configured at design time. Another important aspect is the extended use of non *Commercial Off-The-Shelf* (COTS) hardware when network stack functionality is moved from software to hardware. This leads to higher system costs. Variant 2 can easily be adapted to work with standard components without the need for extra hardware. Such a system would need to prepare a memory buffer that already contains the preconfigured packet headers. The data then needs to be written between the headers, resulting in a complete UDP packet. Compared to the solution discussed in this work, only the DMA configuration would be more complex. Even with a larger overhead per packet, the resulting speeds should easily suffice to support a Fast Ethernet connection with a throughput of 100 Mbit/s. Figure 2.45 shows the structure of the allocated buffer with the

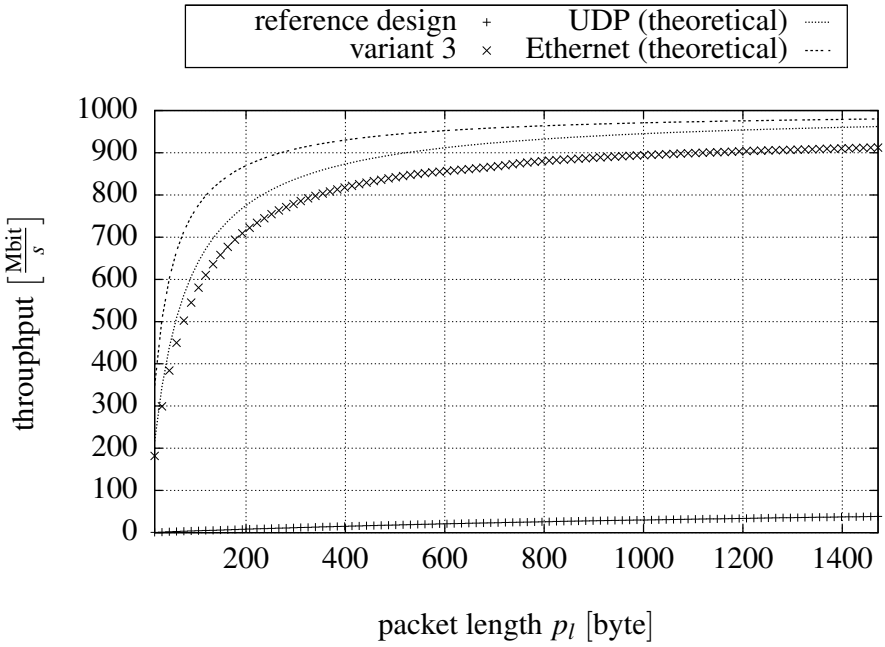


Figure 2.44.: The measured throughputs of the reference design compared to variant 3 with packet generation and transmission in hardware.

predefined headers of the Ethernet frame and the protocols IP and UDP.

Table 2.6 gives an overview of the hardware resource utilization of the FPGA implementation on the Altera development board. The packet generator hardware block of variant 2 consists of 1567 ALUTs and 710 registers. In variant 3, the multiplexer hardware block consumes 156 ALUTs and 124 registers. All values are based on the Stratix II development board from Altera. The variants 2 and 3 require just 4 percent more logic elements than the reference design but they lead to an obvious data rate

Table 2.6.: The FPGA utilization of the different variants.

Criteria	**Variant**			
	0	1a, 1b, and 1c	2	3
logic utilization	39%	39%	43%	43%
combinational ALUTs	12935	12942	14509	14665
dedicated logic register	10817	10804	11514	11638
total registers	11170	11157	11867	11991

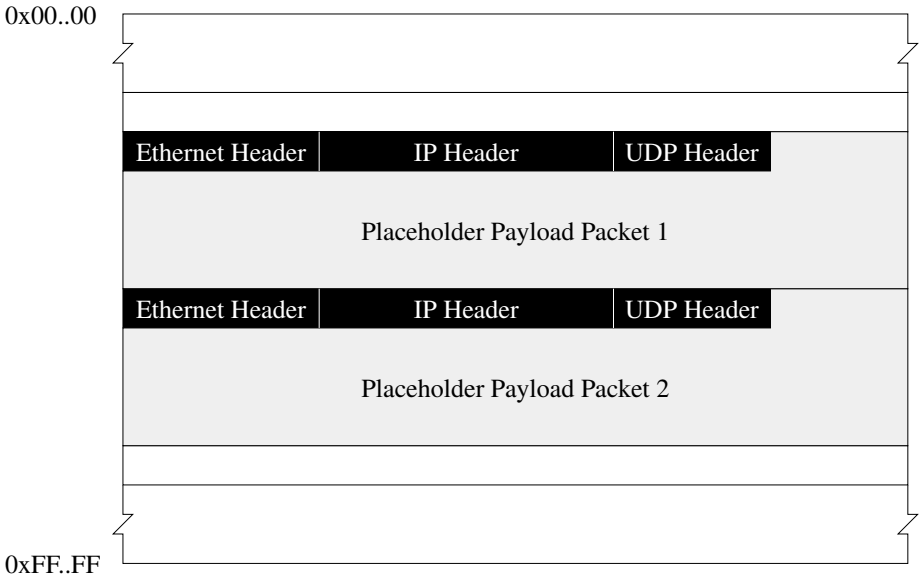


Figure 2.45.: The structure of the allocated memory with predefined headers.

enhancement.

Table 2.7 summarizes several criteria and classifies the investigated transmitter system variants. The goal is to find a solution which fits the application requirements and uses as many COTS components as possible to reduce the overall system costs.

Table 2.7.: The classification of the different system variants related to several criteria.

Criteria	**Variant**					
	0	1a	1b	1c	2	3
throughput utilization	- -	- -	-	-	+	++
error detection capabilities	++	++	-	-	-	-
CPU utilization	- -	- -	-	-	+	++
latency classification	- -	- -	-	-	+	++
COTS rating	++	+	++	+	-	- -
software network stack required	yes				no	
requirements					static design	

2.5. Summary

With regard to Table 2.1 which lists the fundamental work packages to enable Ethernet as an automotive-qualified communication technology, it could be shown by using available custom component technology that error rates, synchronization accuracy, and throughput performance are no general show stoppers for introducing Ethernet. The presented topics discussed general concepts and presented measurements based on specific use cases. Furthermore, it is shown that Ethernet and higher layer protocols offer a wide range of additional features to support future applications. For example, the standardized and widespread transport protocols UPD and TCP which offer port-based addressing and reliable communication.

In the area of error rate analysis, an error rate evaluation concept for Ethernet Hardware PHYs is presented. The goal is to measure the bit and packet error rates as well as the residual error rates for different protocol checksum mechanisms. In addition, the overall bit error distribution and the number of bit errors per packet can be determined. BCI and vehicle measurements show novel bit and packet error results for a given Ethernet PHY. The results offer a first impression of the error behavior of Ethernet communication within the automotive environment. Preliminary measurements show, that no single residual error passed the software network stack at the receiver side. Further longterm vehicle measurements are necessary to produce reliable and reproducible results. Beside the additional measurements, an automated test process could help to simplify the measurements and to automatically generate reports.

In order to fulfill extended automotive temperature requirements, a concept is designed to measure the time synchronization accuracy of distributed nodes under different climate conditions. Industrial climate chambers are used to stress the synchronization end nodes. For the synchronization, the packet-based algorithm specified in IEEE 802.1AS is implemented together with Ethernet as an underlying communication technology. Time synchronization itself is a very sophisticated use case and therefore optimal to test the physical constraints of Ethernet and surrounding parts like oscillators and processors. Exhaustive measurements produced a large set of results and showed that the packet-based IEEE 802.1AS algorithm is able to work under different temperature conditions. A temperature range from $-10\,^{\circ}\mathrm{C}$ ($+14\,\mathrm{F}$) to $+70\,^{\circ}\mathrm{C}$ ($+158\,\mathrm{F}$) is evaluated. Finally, extreme shock temperature measurements

rapidly changed the environment temperature by 80 Kelvin in less than 5 seconds. In summary, IEEE 802.1AS and Ethernet are able to provide high-accurate distributed synchronized timers and synchronization cancellation only appeared a few times during the shock measurements. Note, that the shock measurements today are not required during runtime.

Finally, different concepts are developed to optimize the transmission throughput of an Ethernet-based communication link. The different variants investigated software adaptions as well as hardware support to increase the throughput up to the theoretical limit of Gigabit Ethernet line-rate to support high data rate applications like video- or audio-based applications. The implementation of the variants on an Altera reference board shows that through software optimization (zero-copy and predefined protocol headers), a maximum throughput of about 339 Mbit/s is possible when using a 85 MHz embedded processor. Higher throughputs can be achieved only by additional hardware blocks to release the main processor. The maximum hardware accelerated measured throughput is about 912 Mbit/s. In summary, the maximum throughput greatly depends on the assembled target hardware. The prototype shows a high throughput increase compared to the 38 Mbit/s of the reference design without any optimizations.

3. Integration and Migration Concepts for Ethernet/IP-based E/E-Architectures

Todays premium class vehicles contain up to 80 ECUs which implement hundreds of distributed functions. These functions are either self-developed by the OEM, purchased by a supplier, or a mixed development to use the advantages of both sides. In order to integrate all these functions and their communication relationships, a common E/E-architecture is used by the OEM. This reduces the cabling effort, enables the reuse of components (like sensors or actuators) for different functions, facilitates the testing of interacting functions by having a central entry point, and allows the download of software without having a dedicated flash and diagnostics interface for each component. Present E/E-architectures consist of up to five communication technologies, which are interconnected by a *central gateway* approach as shown in Figure 3.1 to implement and interconnect all distributed functions within the car. Predevelopment activities indicate that the functionality and number of such distributed functions will increase further in the future and this will lead to more complex system requirements for the underlying communication architecture. One reason, for example, is the growth of the communication relationships of complex distributed functions between different communication domains inside the car. A communication domain is introduced to group similar car functions together and to divide the complexity of the overall system in to smaller manageable parts. Each domain is defined by a subset of ECUs interconnected through a subnetwork. Typical domains are *Powertrain*, *Chassis*, *Telematics*, and *Body*. The inter-domain communication

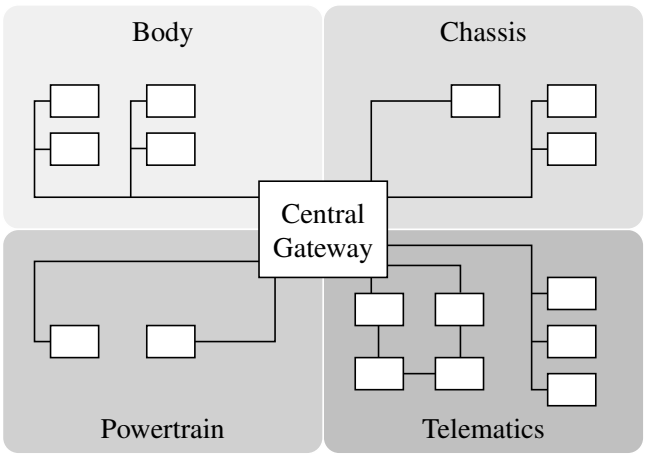


Figure 3.1.: An example of a central gateway E/E-architecture. The ECUs are grouped into the four domains Body, Chassis, Powertrain, and Telematics and inter-domain communication is realized with a central gateway.

is realized by a central gateway which contains multiple communication interfaces connected to the different domains. An application example is the navigation system located in the Telematics domain which uses the current speed value from the Chassis domain to calculate the covered distance of the car within a tunnel, when no signals are available from the satellite-based system. Furthermore, higher data rate demands are caused by an increasing number of video- and object-based driver assistance systems. Next generation cameras will produce high-quality image data which has to be distributed within the car.

These enumerations lead to a high complexity when designing next generation E/E-architectures for future vehicles. Therefore, several ideas exist to design suitable E/E-architectures that will support the requirements of the next decades. One approach is to reconsider the communication paradigm. For example, sending messages in a sporadic on change manner instead of using a cyclic approach just to ensure that the sending device is alive. Instead, this mechanisms can be moved into a separate service. A more complex approach is the integration of an additional new communication technology like Ethernet/IP which can fulfill the upcoming requirements. But to avoid an even higher number of assembled communication technologies, considerations have to be done to efficiently reduce the number of technologies by replacing current technologies, for example. The most difficult idea is to replace the whole central gateway based approach by a *backbone architecture*. Figure 3.2 shows the

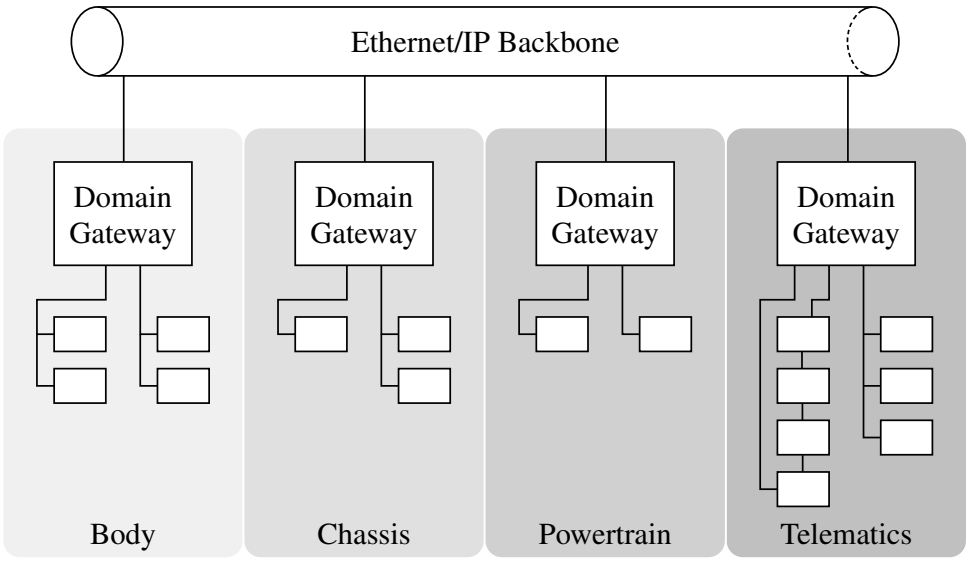


Figure 3.2.: Proposition of a novel Ethernet/IP backbone E/E-architecture. The ECUs are grouped into the four domains Body, Chassis, Powertrain, and Telematics. Inter-domain communication is realized by domain gateways which are connected via a backbone bus.

principle idea of such an architecture. Each domain consists of a domain gateway. Every domain gateway is connected to a high-speed backbone bus which enables the inter-domain communication between different domains. The domain gateways themselves can contain domain-specific functions to process tasks which are used by many ECUs within the domain, for example. In addition, each can handle simple requests from ECUs by itself without causing traffic on the backbone bus.

This chapter addresses the question of how to design Ethernet gateways and presents concepts for replacing current technologies like CAN or FlexRay by Ethernet/IP or AVB to reduce the overall number of assembled communication technologies. In the gateway area, two concepts are proposed to enable CAN and FlexRay to interact with Ethernet/IP and AVB in Section 3.2. In Section 3.2.2, a CAN/Ethernet gateway concept is introduced, implemented, and evaluated by lab and in-vehicle measurements. Different buffer strategies are analyzed to improve the payload ratios of each generated packet. Finally, a full Body-CAN is tunneled via Ethernet/IP within a car to demonstrate the capabilities of such a gateway implementation. For time-sensitive FlexRay communication, a concept is introduced which uses Ethernet AVB to bound the delay and guarantee the throughput (Section 3.2.3). The evaluation presents re-

sults from real communication matrices. Beside the communication from CAN and FlexRay to Ethernet/IP and vice versa, two concepts for replacing CAN and FlexRay by Ethernet/IP in the long-term are finally presented and evaluated by using existing car communication matrices in Section 3.3.

3.1. Fundamentals

This section gives an overview of the current automotive communication technologies CAN and FlexRay, introduces state-of-the-art E/E-architectures and data structures to describe communication, and highlights automotive gateways.

3.1.1. Overview Automotive Communication Technologies

This section provides the necessary basics of CAN and FlexRay to understand the subsequent gateway and migration approaches introducing Ethernet-based communication into the car. In general, CAN represents an event-based communication technology and FlexRay uses an overall time-triggered approach.

CAN In 1981, CAN [CAN91] was developed and standardized by Robert Bosch GmbH to replace the prevalent discrete networking by a common physical infrastructure. Since 1991, the second version is available, and today, almost every vehicle on the road uses CAN for low data rate communication between ECUs. It supports throughputs of up to 1 Mbit/s, utilizes a physically-shared bus topology and uses a priority-based approach for accessing the physical media. The transmitted data is encapsulated within frames and every frame header consists of an identifier which is at the same time the priority for accessing the bus. In general, every connected node will be allowed to start sending a bit-synchronized frame on the bus if the current bus state is idle. The *Carrier Sense Media Access / Collision Resolution* (CSMA/CR) [IEE12b] technique is used to resolve bus access conflicts when two nodes try to start sending frames at the same time. In such a case, the frame with the higher priority wins the arbitration and the node sending the lower priority frame immediately stops its transmission. The maximum supported payload is 8 bytes per frame, but additional transport protocols [ISO11b] are available to enable the transmission of larger payload lengths. In the automotive area, a frame typically contains a *Protocol Data*

Unit (PDU) which itself consists of one or more signals. Signals are the smallest unit for describing information within a car. The payload-ratio in relation to the overall frame ranges between 15 % and 58 % for payload sizes from 1 to 8 bytes. Operational areas of CAN are the connection of low data rate sensors like wheel speed or acceleration sensors or keypads. Table 3.1 summarizes the fundamental differences between CAN and Ethernet.

FlexRay The development of FlexRay [Fle10] started in the year 2000 by the FlexRay Consortium. The technology supports throughputs of up to 10 Mbit/s (single channel) and is based on a time-triggered approach which means that at every point in time, exactly one node is allowed to transmit a frame on the physical shared media. Therefore, no bus access conflicts can arise and no collision detection approach is needed. In general, the communication takes place in *communication cycles* (Figure 3.3). A communication cycle is a predefined time interval consisting of a static and dynamic segment, a symbol window, and the network idle time. FlexRay distinguishes 64 different communication cycles which are transmitted consecutively and continuously. Within the static segment, a TDMA schedule is used to statically assign time intervals exclusively to nodes. This allows a static throughput allocation to nodes at design time. The dynamic segment offers a greater flexibility and allows nodes to dynamically adjust the time interval when having the permission to send a frame. This has the disadvantage, that nodes are not guaranteed to send their frame within a certain dynamic segment. This means, that additional latencies can arise. In

Table 3.1.: Summary of the main differences between CAN and Ethernet.

attribute	CAN	Ethernet
payload layout	PDU → signal	IP → UDP
frame format	base frame format	Ethernet 802.3 frame
header size	47 bit	46 byte
payload size	0 – 64 bit	18 – 1472 byte
addressing	11 bit message identifier	6 byte mac address
topology	shared media	point-to-point
throughputs	125, 500, 1000 kbit/s	10, 100, 1000 Mbit/s
payload-ratio	15 % – 58 %	2 % – 97 % (13 % ≅ 8 byte)

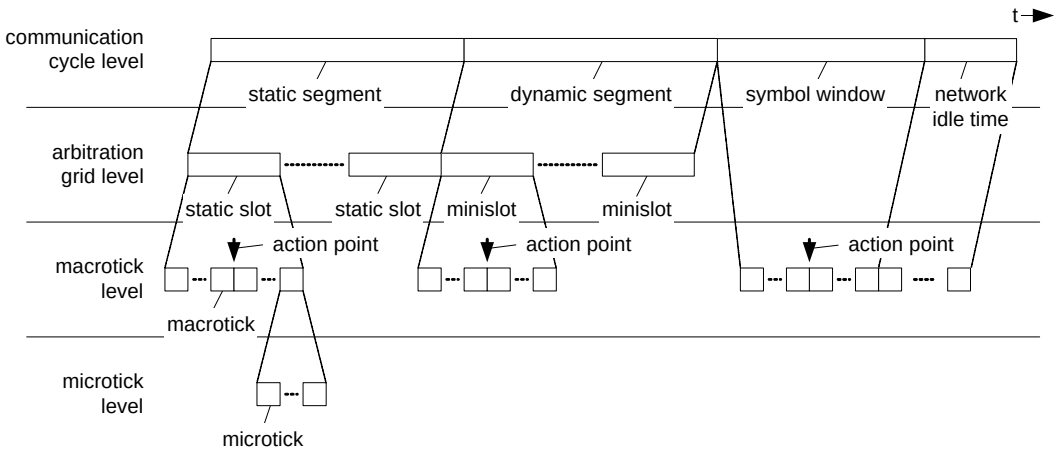


Figure 3.3.: FlexRay communication cycle.

the static segment, a frame is identified by its communication cycle and slot number. Within the dynamic segment, a frame identifier is used to address a frame. The maximum specified payload length is 254 bytes per frame and depends on the designed communication cycle layout. Again, it is possible to use additional transport protocols to support larger payload lengths. The payload layout is similar to CAN, but the payload ratio in relation to the whole frame ranges from 11 % to 96 % for payload lengths between 1 and 254 bytes. Table 3.2 summarizes the fundamental differences between FlexRay and Ethernet.

Table 3.2.: Summary of the main differences between FlexRay and Ethernet.

attribute	**FlexRay**	**Ethernet**
payload layout	PDU $\rightarrow$ signal	IEEE 1722
frame format	FlexRay frame format	Ethernet 802.1q frame
header size	8 byte	46 byte
payload size	0 – 254 byte	18 – 1472 byte
addressing (static)	cycle and slot position	6 byte mac address
addressing (dynamic)	11 bit message identifier	
topology	shared media	point-to-point
throughputs	2.5, 5, 10 Mbit/s	10, 100, 1000 Mbit/s
payload-ratio	11 % – 96 %	2 % – 97 % (13 % $\cong$ 8 byte)

3.1.2. Functional Principle of Credit-based Shaping

The integration and migration concepts for FlexRay will use Ethernet AVB to guarantee the required data rates and latencies within in the Ethernet network. In order to accomplish this, Ethernet AVB uses a *credit-based shaper* to manage the transmit behavior of the different end nodes and switch ports by defining maximum port transmit data rates and additional queuing. Figure 3.4 shows an example of the behavior of a credit-based shaper for one port. The shaper works with an *idle slope* and *send slope*. Therefore, packets can only be transmitted when the credit of the shaper is not negative and no interfering packets from other queues, for example, are send. When transmitting a packet, the send slope will decrement the credit of the shaper. In the opposite case, the credit will be incremented by the idle slope when no packet is transmitted. Remark: Credit values greater than zero can only be achieved by having packets in the output queue which are jammed by interfering packets. With this technique, it is possible to guarantee defined port transmit data rates and minimum transmission intervals for outgoing packets. By choosing proper values for all ports within the system, a reliable network infrastructure can be achieved.

3.1.3. Introduction to Electric and Electronic Architectures

The field of automotive E/E-architecture covers a wide range of tasks including the physical design of hardware and software components like ECUs, cables, connectors, and software as well as the logical design like the partitioning of functions or software components, the definition of communication relationships, and the mapping of system requirements to the respective function blocks. For the physical and logical design, the existence of consistent interfaces is a necessary requirement to ensure the correct functionality of the overall system. In general, different design objectives may be differentiated when designing and optimizing an E/E-architecture: Typical economic objectives are costs, weight, or form sizes of a given system. In the technical area, processor, memory and communication technology resources can be evaluated by simulating or analyzing a given system. Furthermore, it is possible to determine transmission latencies or to test the interaction of different physical or logical components. For example, to check if two interacting components have compatible interfaces. This is typically ensured by common database which

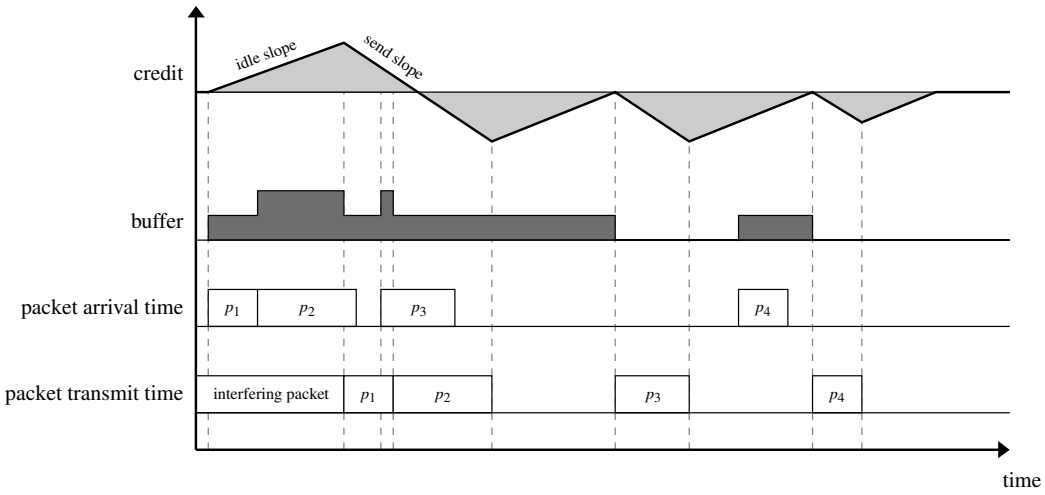


Figure 3.4.: An example of the credit-based shaper used by Ethernet AVB for one port.

contains the whole network communication of one or more vehicles. The description of communication is typically done by standardized exchange file formats like CAN-DBC [Vec12a], FIBEX [ASA11], or AUTOSAR System-Template [AUT11] for communication technologies and OEM-specific file formats for gateway parameter settings. Typically, these files are provided by the OEM and distributed to the different development departments, suppliers, and tool vendors to ensure that the communication interfaces between the different components are well coordinated. For example, there exists one global signal to distribute the vehicle speed within the car. Each ECU within the car requiring this information should then use exactly this signal to receive the vehicle speed information instead of redefining or reconstructing function-specific vehicle speed signals locally.

3.1.4. Automotive Gateways

An automotive gateway is a special kind of ECU consisting of multiple communication interfaces. The main task of the gateway is to interconnect these communication interfaces to enable and control the exchange of information between the different connected communication technologies. Figure 3.5 shows the general concept of an automotive gateway. The internal mapping table defines the mapping between incoming and outgoing telegrams or parts of it. Typically, a gateway must distinguish

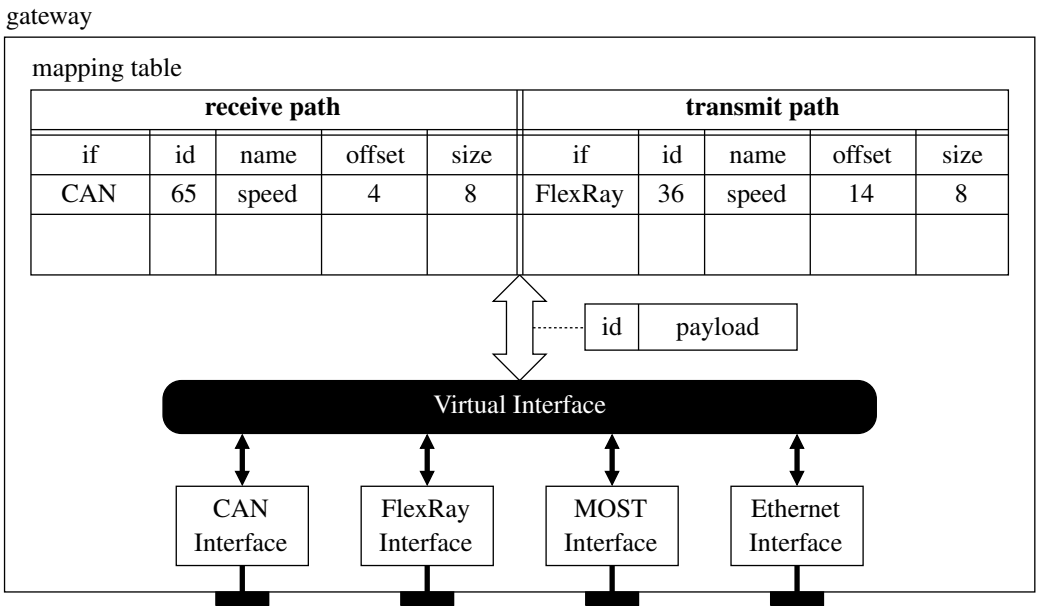


Figure 3.5.: Example of an automotive gateway. For the mapping of data from input ports to output ports, a virtual interface is introduced to hide the hardware-dependent parts of the different communication technologies. A mapping table is defined to describe the routing paths within the gateway.

between *signal-*, PDU-, or *telegram-based* routing. The signal-based approach allows the insertion of individual signals to outgoing telegrams. A forwarding of complete PDUs between different communication technologies is supported by the PDU-based approach. The mapping of complete telegrams – frames or packets including the technology-specific headers – is supported by the telegram-based approach. A virtual interface can be defined to hide the functional mapping layer from the underlying technology-dependent layers. Todays gateways contain CAN, FlexRay, and MOST interfaces. In order to support Ethernet with its higher layer protocols, an extension of the mapping table is needed. In addition, the technology-dependent features have to be adjusted to cooperate with the other communication technologies. An example is to provide the guaranteed throughput of FlexRay and Ethernet AVB. In this case, the routing concept has to consider the differing techniques for throughput allocation, so that the end-to-end link from one ECU connected via FlexRay to the other ECU connected to Ethernet AVB interconnected by the gateway also allows throughput allocation. In general, the throughput and timing requirements must not be violated by

the introduction of a gateway between the transmitter and receiver ECUs. Therefore, special timing analysis has to be performed to verify the communication constraints. Beside the simple routing of just copying memory areas from one buffer to the other as described above, more intelligent mechanisms which also modify the content can be supported. Typical examples are the conversion of physical values into their SI basic units or the adjustment of communication interfaces to enable the insertion of ECUs from other vehicles for example. AUTOSAR-specific gateway requirements are described in [AUT08].

3.2. Ethernet-based Gateway Strategies

This section addresses the question of how to design a CAN/Ethernet gateway and FlexRay/AVB gateway. Detailed concepts are presented, implemented, and evaluated. In addition, the CAN/Ethernet gateway was integrated into a current Mercedes Benz E-class vehicle in order to present real-life performance measurements. In the following, the term *frame* is used to refer a CAN- or FlexRay-frame and the term *packet* refers to an Ethernet/IP-based packet.

3.2.1. Related Work

Most related research addresses the question of how to interconnect Ethernet and CAN communication networks to forward data from one network technology to another. In the automotive area, the *International Organization for Standardization* (ISO) [ISO12] founded the working group *Diagnostics over IP* (DoIP) [ISO11a] which develops a standard for diagnostic communication over IP. They design a standardized vehicle interface which separates in-vehicle communication technologies from external test equipment. The implementation of the interface has to switch data between different in-car networks to the outside IP-based network of the test equipment and is thus working as a gateway. Beside this consideration, there exist multiple concepts and implementations [Sys11] for connecting industrial CAN-based plants to local factory networks. The authors present methods to bundle multiple CAN-frames into IP-packets in a dynamic fashion to reduce the protocol header overhead. In [Mur09], the authors discuss a strategy to migrate from a CAN-network to a FlexRay-based solution. They present a solution to connect to a CAN-based network by using

a special gateway and provide a concept for replacing a complete network through a FlexRay-based communication.

In summary, there exist just a few concepts to forward data from well known automotive communication technologies like CAN or FlexRay to Ethernet. Most solutions are offered as commercial products by industrial Ethernet suppliers. In the case of CAN, these solutions work with dynamic as well as static forwarding mechanisms. Dynamic mechanisms buffer incoming frames into packets in a dynamic fashion. Instead, static approaches use a predefined layout to map the frames into the packets. In general, the internal design is not published.

3.2.2. Controller Area Network

When interconnecting CAN with Ethernet, two significantly different communication concepts meet each other. CAN with its small frame sizes of at most 8 bytes for payload data and Ethernet with its large payloads of up to 1500 bytes. This raises the question, of how to optimize the packaging when routing data from CAN to Ethernet. The following transformation concept presents different variants to optimize the packaging problem. A detailed test setup describes the prototype implementation and the results show measurements for the achievable Ethernet throughputs and resource utilizations based on different input scenarios including stress tests.

3.2.2.1. Transformation Concept

The data flow between the different gateway components can be divided into traffic from CAN to Ethernet and traffic from Ethernet to CAN. Figure 3.6 displays the flow from CAN to Ethernet. After a frame is being received by the CAN-controller, it is stored in an internal receive buffer and an interrupt is generated to the gateway processor. The dispatcher process then reads the frame and passes it into an internal queue of the gateway, stored in its local memory. When the internal queue is ready to be processed – this depends on the applied transformation variant – the contained entries are passed to the software network stack of the gateway procedure which generates the packet and passes it to the transmit queue of the Ethernet-controller. In contrast to this, the data flow from Ethernet to CAN is shown in Figure 3.7. After a packet is being received and stored by the Ethernet-controller, an interrupt is generated to activate the software network stack which processes the packet to gain the encapsulated

Figure 3.6.: The data flow through the different involved hardware and software components when forwarding data from CAN to Ethernet. After receiving a frame by a CAN controller, the frame is stored in the internal memory and an interrupt is generated. A receive and dispatch task then forwards the packet to right outgoing queue depending on a mapping table. Every time a send event occurs, the network task is activated to generate and append the protocol headers and to forward the assembled packet to the send queue of the Ethernet controller.

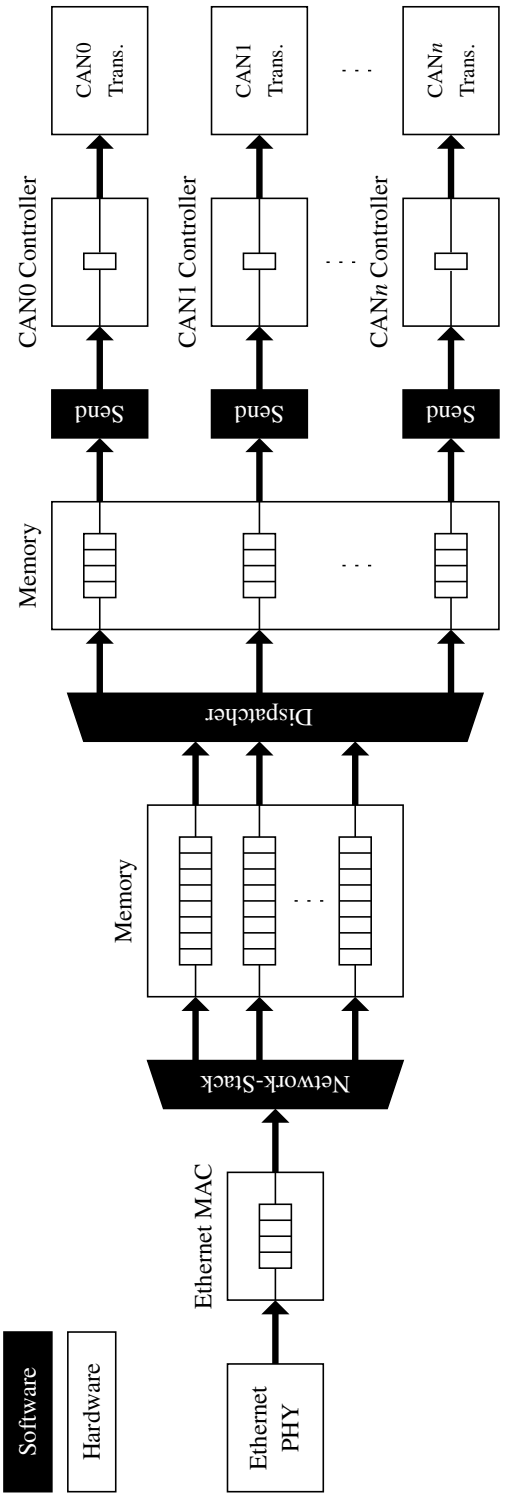


Figure 3.7.: The involved hardware and software blocks when forwarding data from Ethernet to CAN. After receiving a packet by the Ethernet controller, the packet is stored in internal memory and an interrupt is generated. The network stack task then processes the packet and copies the payload into the internal memory. A dispatch task is then used to divide and forward the data to the right output queue dependent on a mapping table. Finally, the send function is used to initiate the send process.

frames. The frames are stored in an internal queue and a second software process is triggered which dispatches the frames to the outgoing queues. Each outgoing queue is processed by the communication controller which checks if the transmit buffer of the CAN-controller is free and then it passes the frame from the outgoing queue to the transmit buffer of the CAN-controller. Based on this frame or packet processing, different variants are observed for the queue handling when sending data from CAN to Ethernet. In the following, four different conversion variants as shown in Table 3.3 are discussed.

Variant 0: Reference Design This variant is used as a reference for the subsequent variants and applies a one-to-one mapping to encapsulate frames. This means that every received frame is encapsulated in a dedicated packet. A huge protocol header overhead, a large number of packets, and a high resource utilization are some of the disadvantages of this straightforward approach. The advantages are a low jitter of the packets and small additional latency offsets. An example of the one-to-one mapping is shown in Figure 3.8(a) where each frame is encapsulated in a dedicated packet. It shows a number of received frames and the related packets which have been created over time.

Variant 1: Buffered Approach A buffered variant is introduced to reduce the protocol header overhead caused by Ethernet, IP, and higher protocol headers. Therefore, incoming frames are queued in an internal buffer to group multiple frames together before creating one packet. The transmission of a packet is triggered by one of the following events:

- **buffer full**
  Each buffer b has a buffer size b_s that specifies the number of frames which can

Table 3.3.: Four considered transformation variants.

Variant	0 (reference)	1	2	3
Method	1:1 transformation	n:1 transformation		
	one-to-one mapping	buffered approach	timed approach	urgency approach

be stored in it. If the buffer is full, the transmission will be initiated immediately.

- **timeout**
  Each buffer b also has buffer timeout b_t that specifies a maximum period, before the transmission is initiated automatically, even if the buffer is full or not.

The advantages of this approach are lower protocol header overheads, a smaller number of packets sent on the network, and lower resource utilizations. On the other hand, the buffering may introduce additional latency and cause higher jitters. Figure 3.8(b) shows an example of a buffered approach where five packets are being created, three of them are timeout-driven and the two others are caused by a full buffer. In addition, the timer and buffer fill level are shown over time.

Variant 2: Timed Approach Variant 2 extends the buffered approach by taking into account that CAN uses priorities for scheduling frames on the shared bus. The idea is to dynamically reduce a timer value when a high priority frame arrives to shorten the buffering time of high priority frames. It is also possible to use a staged approach. This means, that several priority classes exist which have different values for shortening the timer of a buffer. Advantages of this method compared to variant 1 are that it is possible to support priority-based transmissions and this will result in smaller latency and smaller jitter values for the processed frames. The dynamic reduction of the timer creates more packets per time unit and causes higher protocol header overheads compared to variant 1. Figure 3.9 demonstrates an example for this timed approach. The frames with the priorities 1, 2, and 3 lead to an additional decrement of the timer of two and the frames with the priority 4, 5, and 6 lead to an additional decrement of the timer by one. All other messages do not influence the timer.

Variant 3: Urgency Approach The urgency-based approach is an extension of the timed approach. In this case, high priority messages are being sent immediately. This leads to an additional event:

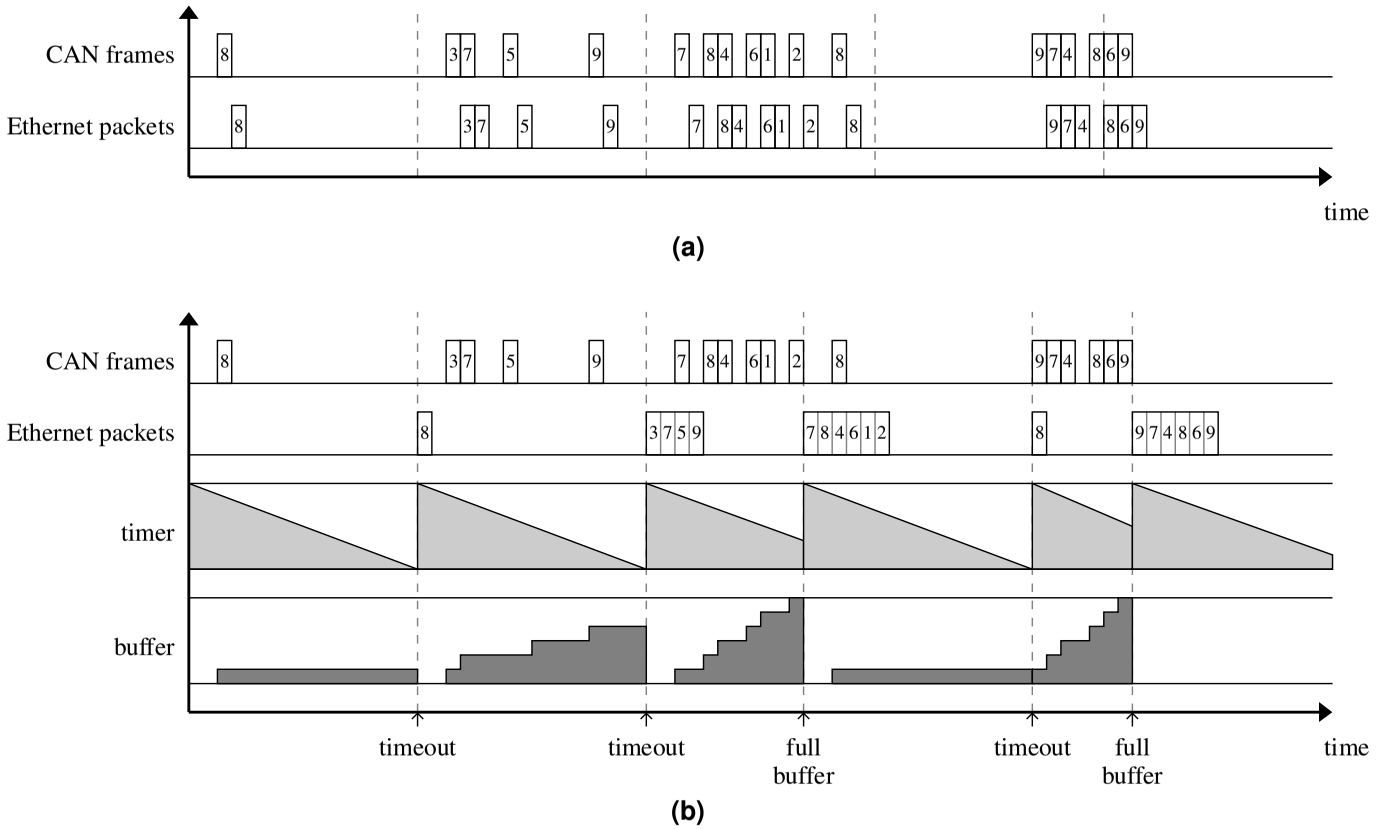


Figure 3.8.: Example for the reference design and the buffered approach for CAN frame and Ethernet packet conversion and transmission: (a) Shows the transformation of incoming frames by a one-to-one mapping. (b) Presents the idea of a buffered approach. The transmission of a packet is either triggered by a full buffer or a timeout. Note that the width of the frames and packets does not correspond to the actual frame oder packet size.

Figure 3.9.: An example for the timed approach for CAN frame to Ethernet packet conversion and transmission: Different identifier groups are used to influence the timer. For example, high priority messages will lead to a higher decrement of the timer. Note that the width of the frames and packets does not correspond to the actual frame oder packet size.

- **instant send**
  Whenever a frame with a high priority is received, the transmission of the packet is triggered immediately.

It is expected that the additional latencies and the jitter values of the frames are further reduced. Figure 3.10 shows an example of the urgency approach. In this case, the frames with the priorities 1, 2, and 3 lead to the instant sending of the packet.

3.2.2.2. Test Setup

The following test setup for evaluating the four introduced common strategies consists of two CAN/Ethernet-gateways which are interconnected through a switch. Furthermore, both gateways are connected via a CAN-interface card to a desktop computer and the computer is also connected to the switch. A test run consists of the execution of a trace of frames which is sent to the first gateway. Then, the first gateway maps this traffic to packets and sends these packets to the second gateway via a switch. The second gateway maps the received packets to the CAN-interface and sends the frames back to the computer. During the benchmark run, the send times, the switched packets, and the arrival times are recorded for post processing. The send and receive CAN-interfaces of the computer are time-synchronized. Figure 3.11 shows the architecture of the implemented CAN/Ethernet-Gateway. The gateway itself has six CAN-interfaces and one Ethernet-interface. The Altera Stratix II FPGA is programmed using a Nios II *System on Chip* (SoC) [Alt09] processor running at 85 MHz, an SRAM Controller connected to a 2 MB SRAM device, and the network controllers. All hardware components are connected via an Avalon Bus. As an operating system, the MicroC/OS-II including an Iniche TCP/IP network-stack is applied.

3.2.2.3. Experimental Results

The experimental results are based on vehicle-representative CAN network traces using different types of messages like cyclic or spontaneous messages with different priorities and sizes as well as a real vehicle network trace of a BODY-CAN from an up-to-date type series. For all introduced variants, a) the measured end-to-end communication latency, b) the resulting average Ethernet/IP throughputs, c) the jitter

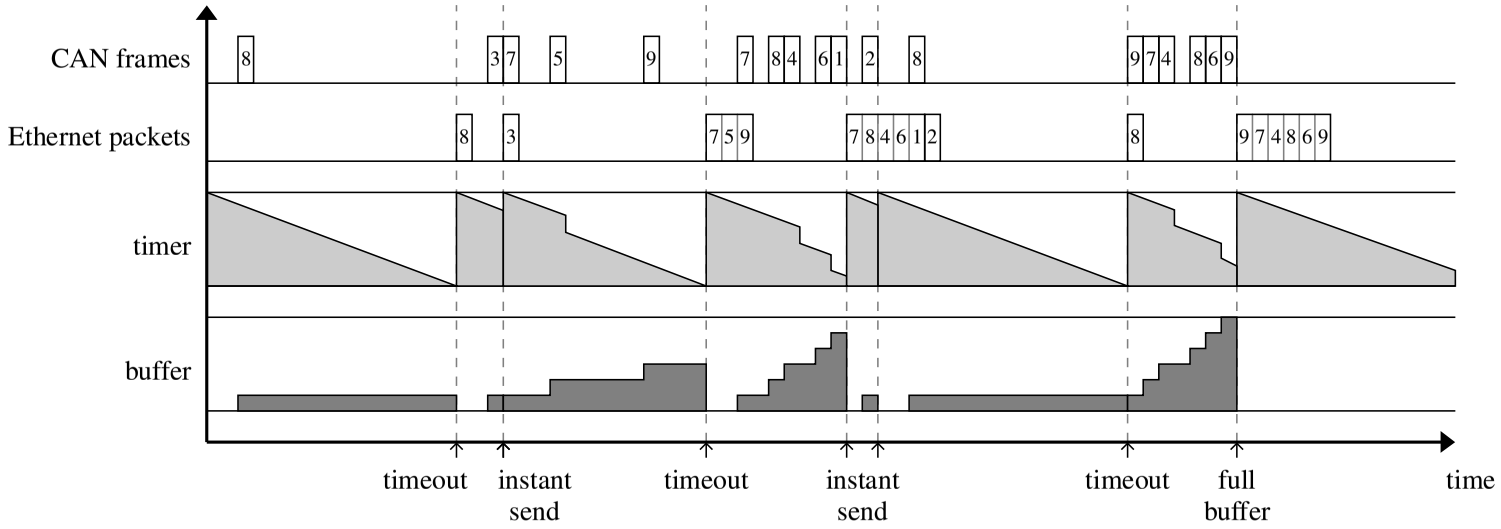


Figure 3.10.: The urgency-based approach for CAN frame to Ethernet packet conversion: The transmission of a packet is triggered by one out of three events. A full buffer, a timeout, or a high priority identifier. Note that the width of the frames and packets does not correspond to the actual frame oder packet size.

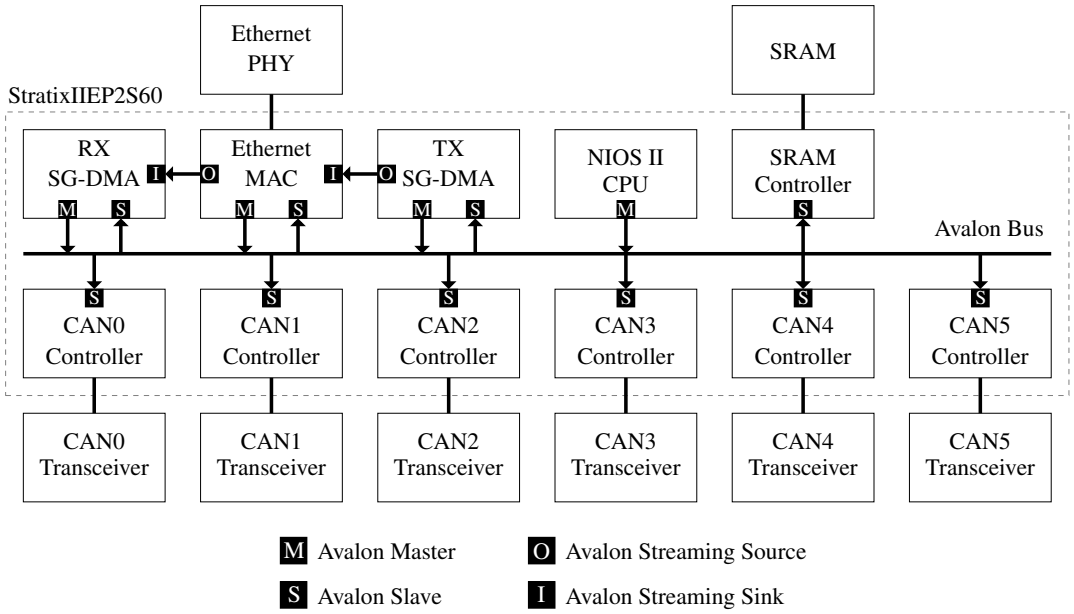


Figure 3.11.: The architecture of the implemented CAN/Ethernet-gateway based on an Altera development board.

variation of the frames, and d) the resource utilization of the gateway processor are summarized.

End-to-End Latency The end-to-end latency represents the round-trip time from sending a frame by the computer to receiving the same frame back from the second gateway at the computer. Figure 3.12 shows the measured average end-to-end latencies for a 500 kbit/s CAN network. As can be seen, the average latency amounts to 500 µs when processing frames one-by-one. The presented buffered, timed, and urgency approaches use a buffer size of 20 frames and a timeout value of 5 ms. This results in higher average latencies because of the queuing of frames within the gateway. The latencies of the timed and urgency approach are lower compared to the buffered approach caused by higher trigger rates of the buffers due to the support of priorities and the *instant send* event. Figure 3.13 shows the average end-to-end latencies of the buffered approach for different buffer sizes b_s and buffer timeout values b_t. The missing reading points result from a premature *timeout* or *buffer full* event. In these cases, the events occurred before enough frames arrived within in a given time interval or too many frames arrived due to a high bus load.

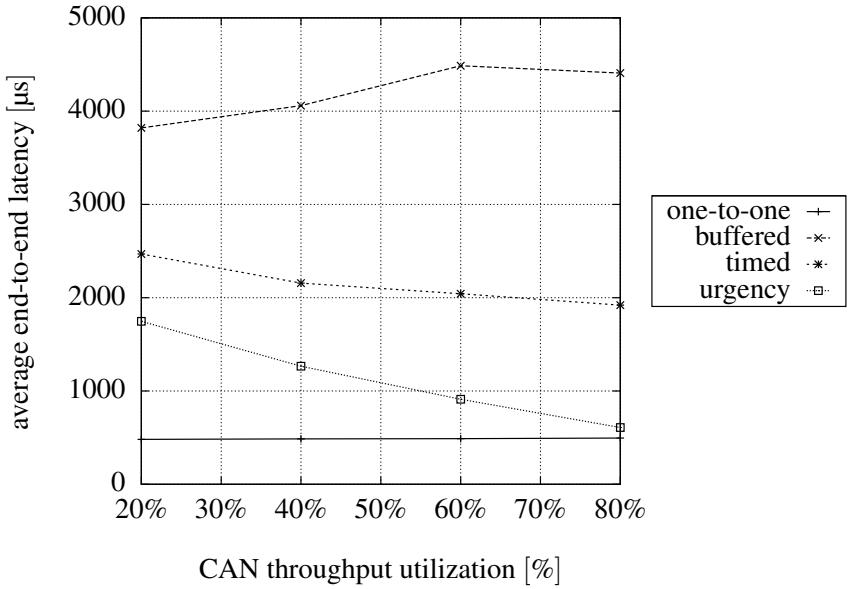
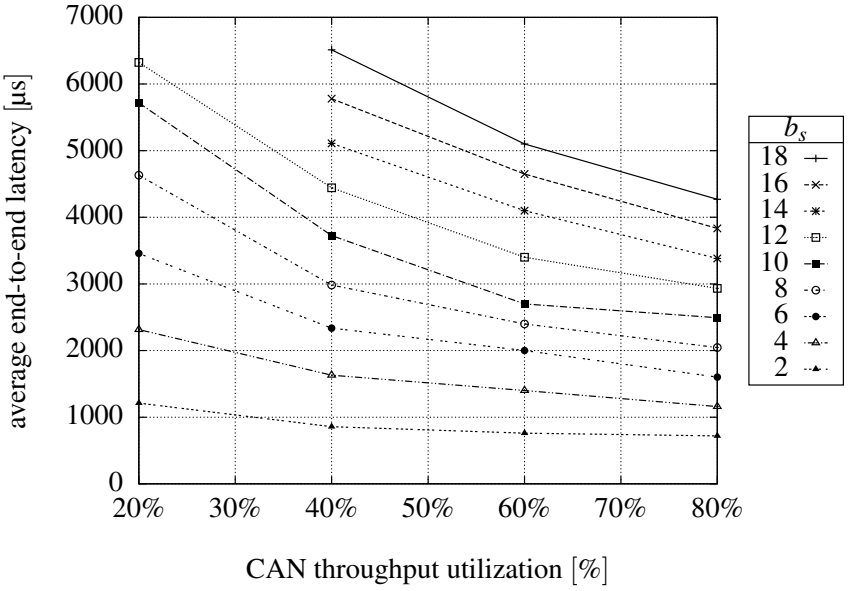


Figure 3.12.: The measured average end-to-end latency of all variants for a 500 kbit/s CAN network.

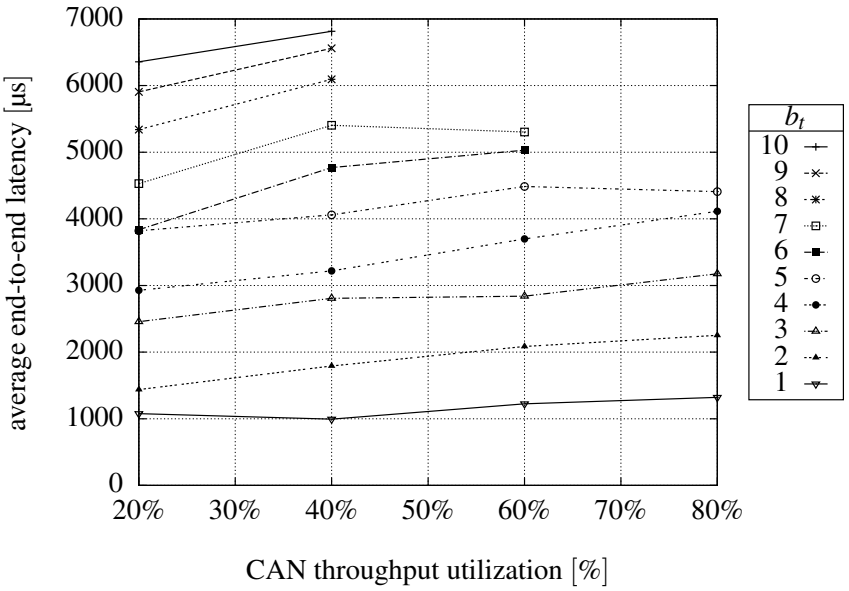
Ethernet Throughput Figure 3.14 presents the measured average throughputs of the four variants compared to a CAN network for a utilization of 40 % which is a typical value for automotive buses. In the case of Ethernet/IP, we used the User Datagram Protocol (UDP) [Pos80] as a transport protocol. The one-to-one mapping produces the largest throughput of 1.18 Mbit/s compared to the other variants due to the mapping of one frame to one packet. In contrast, the buffered approach causes the lowest throughput ranging from 0.3 to 0.7 Mbit/s for different buffer timeouts b_t. Assuming the one-to-one mapping and a fixed CAN payload length of 8 byte, the theoretical minimum Ethernet throughput $b^{min}_{Ethernet}$ can be calculated by Equation (3.1):

$$b^{min}_{Ethernet}(b_{CAN}) = \frac{512}{108} \cdot b_{CAN} \tag{3.1}$$

Resource Utilization In parallel with the timing measurements, we also traced the resource utilization of the gateway processor. Figure 3.15 shows the average processor utilization of the 85 MHz SoC which varies from 4 % to 26 % depending on the applied variant. Most of the processing resources are utilized by the software



(a)



(b)

Figure 3.13.: The average end-to-end latencies of the buffered approach for different buffer sizes b_s [frames] and buffer timeout values b_t [ms].

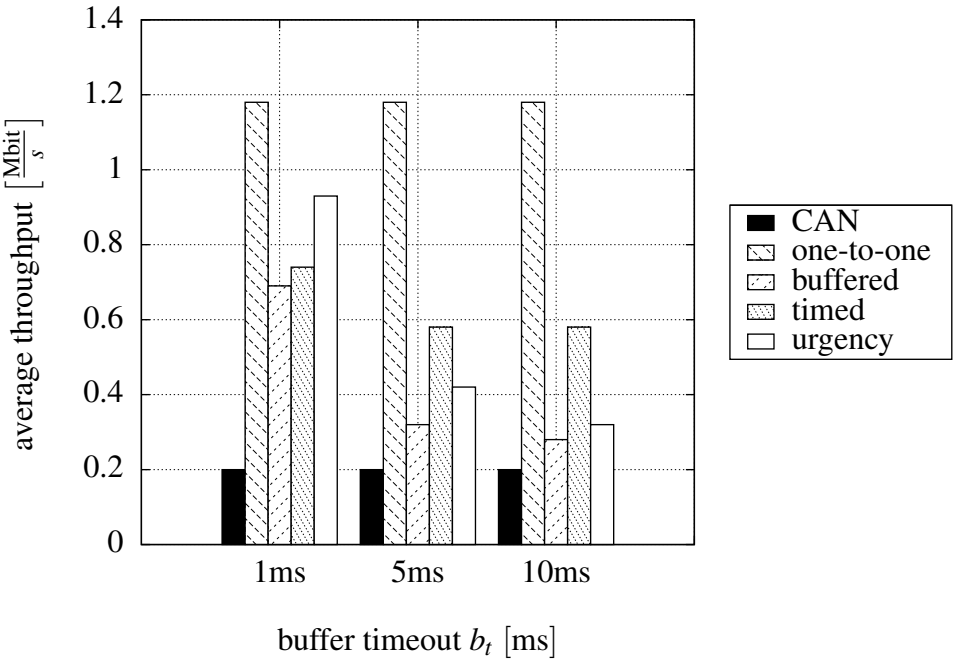


Figure 3.14.: The measured average throughput of all variants for a 500 kbit/s CAN network.

network stack which prepares the packet headers of the protocols UDP, IP, and the Ethernet frame format.

In-Vehicle Measurements The results obtained from the Mercedes E-class network configuration are finally presented in Figure 3.16. We explored a BODY-CAN network with a bus speed of 125 kbit/s and a utilization of about 45 %. The average end-to-end latency differs from 1.8 ms – in the case of a one-to-one mapping – to 6.8 ms when the buffered approach is used. The setup tunneled the whole BODY-CAN network traffic via two gateways.

Table 3.4 summarizes the advantages and disadvantages of the different observed variants. The packet overhead could be optimized by introducing buffering and additional events which reduce the backlog time of packets. This resulted in a smaller number of packets per time and a reduced resource utilization of the gateway processor. Disadvantages of the buffered approaches are the higher latencies and jitters of frames.

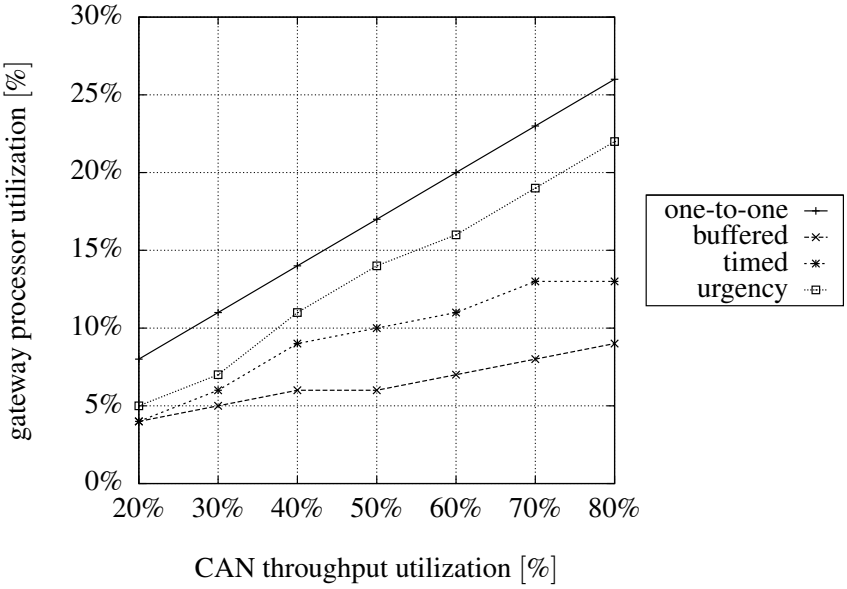


Figure 3.15.: The average resource utilization of the gateway processor.

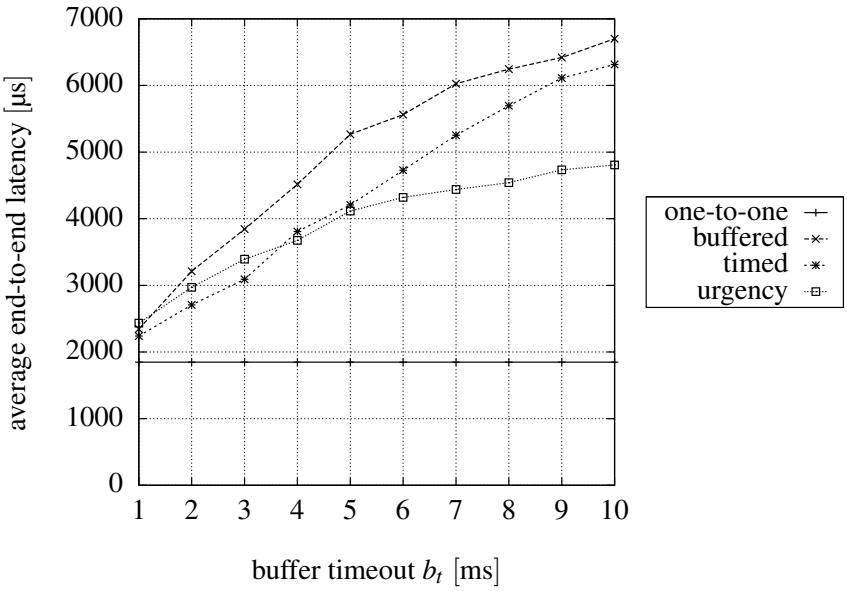


Figure 3.16.: The average end-to-end latency of a BODY-CAN network with a transmission rate of 125 kbit/s (utilization of about 45 %).

3.2.3. FlexRay

In theory, FlexRay offers a huge number of parameters to adopt the FlexRay communication cycle to application-specific needs. This results in an increased complexity when designing the optimal communication schedule which fits to all applications connected to the FlexRay bus. Other constraints like the possibility of integrating ECUs in different car lines also restrict the degree of freedom when creating a communication schedule. In practice, most OEMs committed to use a five millisecond communication cycle to restrict the overall complexity. When utilizing 50 % percent for the static segment and 50 % for the dynamic segment, the static segment will have a length of 2.5 ms. In the case of 60 static slots, for example, the maximum theoretical frame length is about 40 bytes. The header and the frame offset of each frame within a static slot will further decrease the payload per slot. The small payload sizes rises the question of how to optimize the packaging of the FlexRay frames into Ethernet packets when designing a FlexRay to Ethernet gateway. In addition, FlexRay uses a TDMA schedule to guarantee a fixed time slot per communication cycle for an ECU. This allows the reservation of guaranteed data rates and defines upper bound communication latencies. The following concept will optimize the packaging problem of the static segment of FlexRay and utilizes Ethernet AVB [IEE12a] to guarantee predefined bandwidths and worst case communication latencies.

Table 3.4.: The classification of the different transformation variants related to several criteria.

Criteria	**Variant**			
	0	1	2	3
packet overhead	- -	++	+	-
CPU utilization	- -	++	+	-
latency	++	- -	-	+
jitter	++	+	-	+
burst classification	++	- -	-	+

3.2.3.1. Transformation Concept

Again, the transformation concept is divided into the two routing paths from FlexRay to Ethernet AVB and vice versa. Figure 3.17 presents the necessary hardware and software blocks to forward information received from Ethernet packets to FlexRay frames. After a packet is received by the communication controller, the AVB network stack interprets the packet and copies the payload data into the memory. The payload can consist of different streams and a stream includes different samples which correlates to FlexRay frames. In the next step, a dispatcher function forwards the entries to different queues at the egress ports of the different connected FlexRay controllers. The FlexRay controller itself triggers the sending of the FlexRay frames. The more interesting case is shown in Figure 3.18 where FlexRay frames have to be collected and inserted into Ethernet packets. After a FlexRay frame is being received by the FlexRay controller, a software receive and dispatch function is evolved to forward the received frame to the corresponding queue. Each queue represents a stream of FlexRay packets (also called samples) between the gateway and one end node within the Ethernet network. A statically predefined buffer table determines the transmission time of packets. Whenever a packet is ready for being transmitted, the AVB network stack creates the necessary packet headers and copies the packet to the AVB controller which includes a traffic shaping mechanism and finally transmits the packet. Based on this frame or packet processing, different buffer strategies are introduced to optimize the packaging when sending data from FlexRay to AVB. Since the schedule of the static segment is known at design time, the calculation of the different buffer strategies can also be done at design time by analyzing the schedule table. Since no FlexRay/AVB gateways are available today, the following concept is described by dividing an actual FlexRay configuration into two parts as shown in Figure 3.19. The set of connected FlexRay ECUs is therefore split into two subsets. The first subset is connected via a first FlexRay to the first gateway 1 and the second subset is connected via a second FlexRay to the second gateway 2. On the Ethernet AVB side, both gateways are interconnected via a switch. The general steps to determine the packaging of frames into packets for the different variants is shown in Figure 3.20 and presented in the following.

Figure 3.17.: The involved hardware and software blocks to forward data from Ethernet AVB to FlexRay. After receiving a packet by the AVB controller, the packet is stored in the internal memory and an interrupt is generated. The AVB network stack then interprets the packet with its FlexRay samples and copies the samples into internal memory. A stream dispatch task is then used to copy the samples which include the FlexRay frames to memory where the FlexRay controller is assigned to. The FlexRay controller is preconfigured at design time and triggers the sending of the FlexRay frames by itself.

Figure 3.18.: The data flow through the different involved hardware and software components to forward data from FlexRay to Ethernet AVB. After receiving a frame by the controller, the frame is stored in the internal memory and an interrupt is generated. A receive and dispatch software task then forwards the packet to the right outgoing queue depending on a mapping table. The send condition of each stream is defined at design time. After a packet is ready for being transmitted, the AVB network task is activated to generate the protocol headers and to forward the packet to the send queue of the AVB controller.

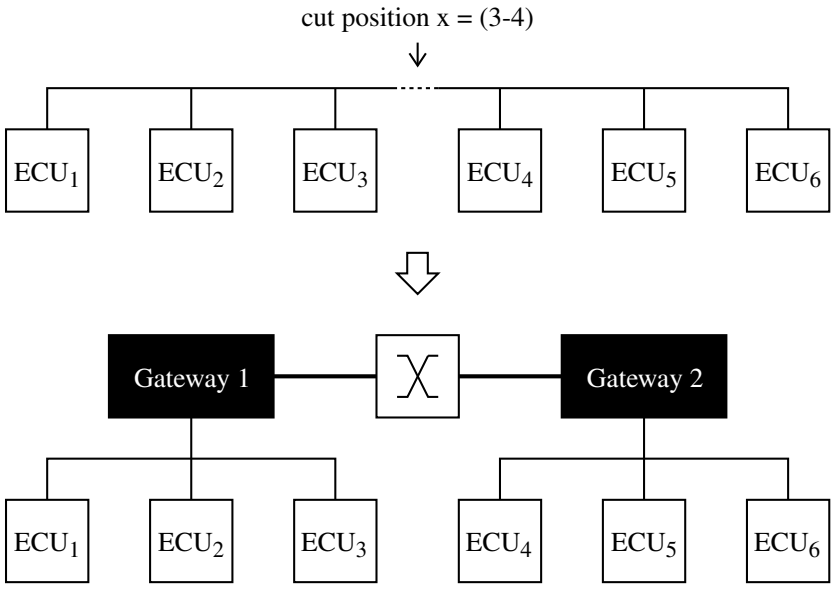


Figure 3.19.: Assumed system topology. An actual FlexRay network is split into two subnetworks which are interconnected by two gateways.

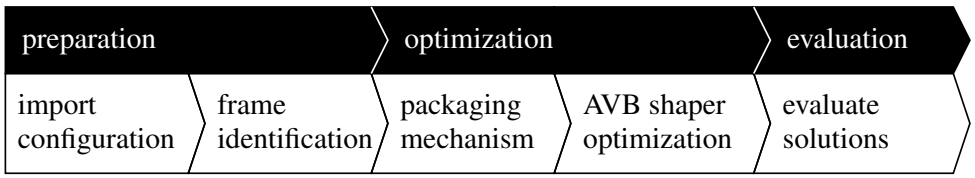


Figure 3.20.: The necessary steps of the transformation concept to perform the mapping from FlexRay frames to Ethernet AVB packets and vice versa.

Configuration Import The first step of the transformation concept is the import of a FlexRay configuration from an AUTOSAR System Template [AUT11] file based on XML. The import covers the following necessary information:

- $t_c \in \mathbb{N}$ – the communication cycle length t_c in microseconds.
- $t_s \in \mathbb{N}$ – the length t_s of the static segment in microseconds.
- $n_s \in \mathbb{N}$ – the number n_s of static slots within the static segment.
- $e \in E$ – where E denotes the set of connected ECUs to the FlexRay bus.
- $f \in F$ – where F denotes the set of frames sent over the FlexRay bus.

communication cycle 0 – 6	static segment 1 – 8 (frame$_{ECU}$)								dynamic segment
	a_1	b_6	c_2	d_3		g_4	h_5		
		b_6	e_3			g_4		i_2	
	a_1	b_6	c_2	d_3	f_1			j_6	
		b_6	e_3			g_4	h_5		
	a_1	b_6	c_2	d_3		g_4		i_2	
		b_6	e_3		f_1			j_6	
	a_1	b_6	c_2	d_3		g_4	h_5		

Figure 3.21.: A sample configuration of a FlexRay network.

- $l_p \in \mathbb{N}$ – the maximum payload length l_p of a frame in the static segment.
- $\mathbf{M} : (f, e, R \subseteq E)^{64 \times n_s}$ – communication matrix $\mathbf{M}$. Each entry contains the transmitted frame f by its corresponding ECU e and the set of receiving ECUs R for each static slot over all 64 different possible communication cycles.

All of the following examples used to describe the transformation concept are based on the configuration presented in Figure 3.21. It consists of a communication cycle with a length of $t_c = 5\,\text{ms}$, a static segment length of $t_s = 3\,\text{ms}$, $n_s = 8$ static slots, and a maximum payload length of $l_p = 26$ bytes. Let six ECUs be connected to the FlexRay ($E = \{e_1, ..., e_6\}$) and a total number of ten frames ($F = \{f_1, ..., f_{10}\}$) sent during the 64 communication cycles. Frame f_a which is sent by ECU e_1 has a cycle time of 10 ms, for example.

Frame Identification After a configuration has been imported, the frame identification is done. For this purpose, each frame that has to pass a gateway is associated with a *stream*. A stream represent a flow of data from the gateway to one receiving Ethernet end node. Frames which are received by more than one end node are mapped to multiple streams related to their receivers. Alternatively, multicast streams can be used to avoid the retransmission of identical frames in different packets. Note, in this case, an end node can also be another gateway. The example in Figure 3.22 maps the four frames f_a, f_c, f_d, and f_i from the first FlexRay network to one single stream s_1 and all frames f_b, f_g, and f_h of the second FlexRay network to another stream s_2. Frames which remain in their respective network f_e, f_f, and f_j are not mapped to a

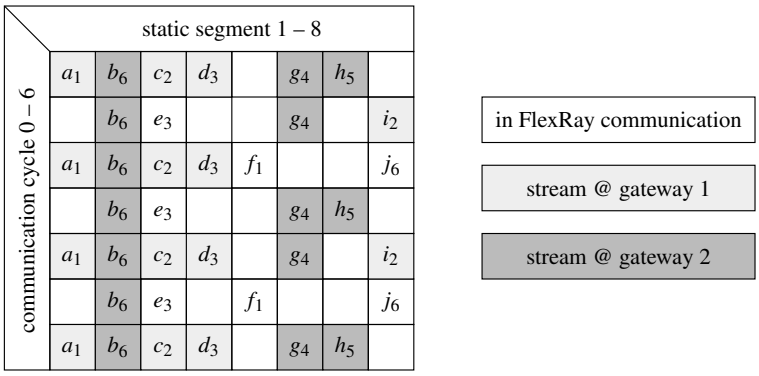


Figure 3.22.: The mapping of FlexRay frames to corresponding Ethernet AVB streams.

particular stream.

Packaging Optimization Packaging optimization deals with the problem of mapping frames to packets and introducing the packet structure. Figure 3.23 shows a chosen packet structure for the packets created by the gateway that is based on the standard IEEE 1722 and uses a new automotive subtype. A proposal for such an automotive subtype in IEEE 1722 was presented at [KSG+11•]. Figure 3.23a describes the IEEE 1722 header. The *subtype* field carries the new automotive subtype identifier which influences the interpretation of the header fields marked with a gray background. The *avtp timestamp* field represents the time of measurement of the first sample within the payload. Note that in the case of FlexRay, all other timestamps of the contained frames can be calculated since the layout of the frame mapping is static and created at design time. The last three fields *protocol*, *extension*, and *number of frames* describe the protocol within the payload (FlexRay, for example), the byte offset within the payload before the first sample starts, and the total number of samples within the payload. Communication technology specific information can be inserted within the extension section at the beginning of the payload before the first sample starts. Each packet can only contain samples from one communication technology. Figure 3.23b shows the packet structure of one sample for FlexRay. In this case, the entire header and payload of the FlexRay frame represents one sample. Table 3.5 gives an overview of the three different mapping approaches *one-to-one*, *in-cycle*, and *multiple-cycle* which are described in the following.

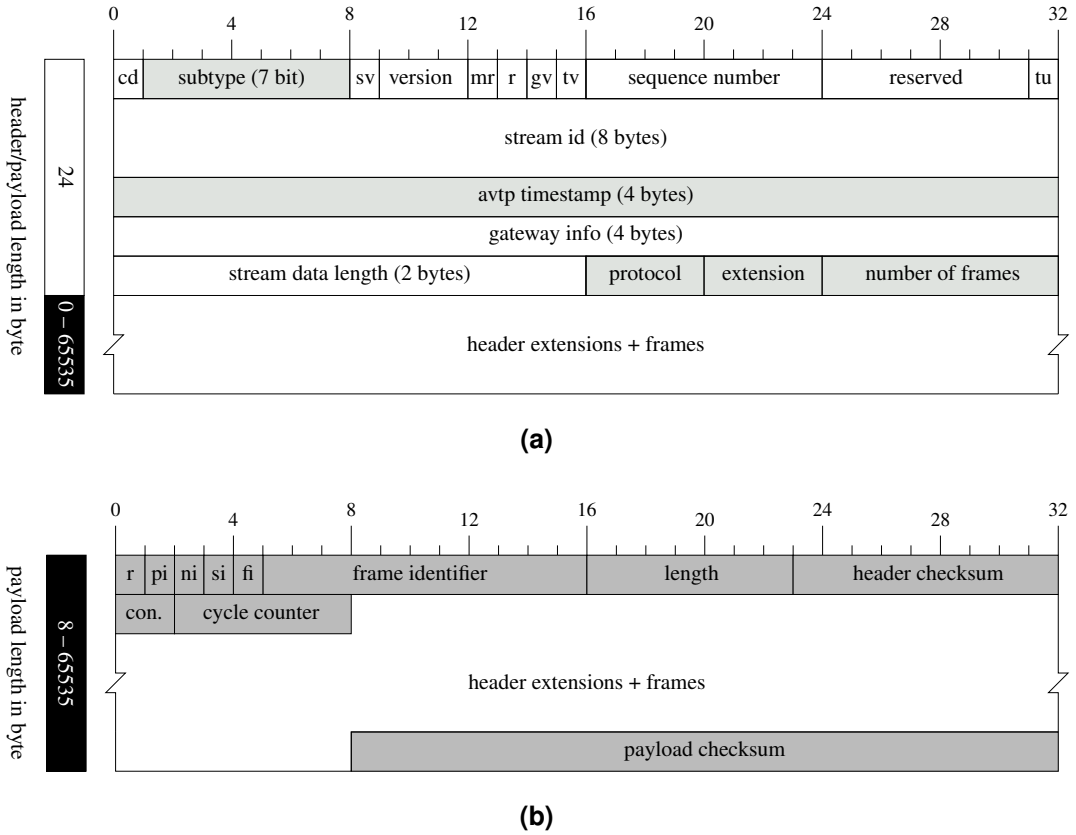


Figure 3.23.: All packets created by the gateway are based on IEEE 1722 and use a special subtype for describing automotive specific data. (a) Describes the adjusted IEEE1722 header. All subtype-related fields are highlighted. (b) Shows the payload of the IEEE 1722 automotive subtype packet. In this case, a FlexRay frame.

Reference – One-to-One Mapping The one-to-one conversion is considered as a reference for the following mapping concepts. In this case, each FlexRay frame is mapped one-by-one to a IEEE 1722 packet. The advantages are no additional buffer latencies and minimal jitter values due to the almost non-existent queuing latencies.

Table 3.5.: The considered transformation variants of the FlexRay/AVB gateway.

Variant	0 (reference)	1	2
Method	1:1 transformation	n:1 transformation	
	one-to-one mapping	in-cycle approach	multiple-cycle approach

The main disadvantage of this approach is the inefficient mapping of just one relatively small frame into one large packet. Figure 3.24(a) shows an example where all gateway relevant frames are mapped into their own packets. The result is a set of 4 packets for gateway 1 and a set of 3 packets for gateway 2.

Variant 1 – In-Cycle Solution This approach tries to optimize the packaging overhead by grouping several frames f_1, f_2, ... , $f_n \in F$ of the same communication cycle together in one packet p. The necessary conditions are:

- All frames in F have the same receiver or group of receivers in the case of multicast addressing.
- Despite the additional queuing latencies for each frame $f \in F$, a given end-to-end deadline for each frame f has to be met.
- The number of grouped frames have to be smaller than a given maximum buffer size s_b.
- If there are several equal solutions for the mapping of frames to packets of one communication cycle, the solution with the minimum buffer overhead has to be taken.

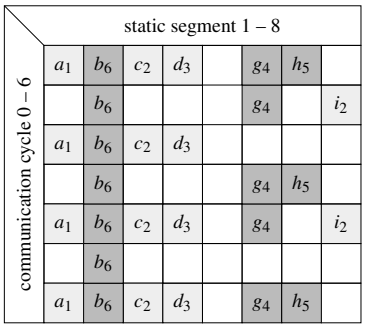
The main advantage is the more efficient packaging of small frames into packets and the reduced number of packets which have to be generated, sent, and received. The disadvantages are higher end-to-end latencies due to the additional queuing latencies and higher jitter values. An example for a buffer size of $b_s = 2$ is shown in Figure 3.24(b). At gateway 1, the frames f_c and f_d can be grouped together and sent within packet p_B. The remaining frames f_a and f_i are mapped to their own packets. At gateway 2, every fourth communication cycle, the frames f_g and f_h are grouped together and sent within packet p_B.

Variant 2 – Multiple-Cycle Solution FlexRay applies transport protocols to send data structures which are larger than the maximum supported payload length. The multiple-cycle approach is introduced here to map FlexRay transport protocols efficiently to large IEEE 1722 packets. The data structures are therefore split into several FlexRay frames. To reassemble the FlexRay frames at the receiver nodes,

additional information is stored within each frame. When using transport protocols, the gateway groups several transport protocol frames together and send the whole information within large packets to the receiving nodes. The main advantage is, that the large *Maximum Transmission Unit* (MTU) of Ethernet – the maximum Ethernet frame length – can be used to get rid of the transport protocol. Figure 3.24(c) shows the extension of the in-cycle approach by grouping the frames f_b together. In this case, it is necessary to know, that f_b carries information based on a transport protocol.

AVB Shaper Configuration Beside the optimization of the packaging, the reservation of the required data rate has to be done. Therefore, the AVB shaper has to be configured with the required data rate. By knowing the communication matrix $\mathbf{M}$ and the packaging of the frames into packets, the necessary data rate can be calculated for all communication cycles. Since the data rate allocation is defined in bits per second, two different approaches are possible: 1) The reserved data rate can be set to the exact calculated amount of data over one second. But this could result in additional queuing latencies within the AVB shaper. That is the case when the amount of data within one specific cycle is higher then the average amount of data calculated by the overall data within one second divided by the number of cycles. To avoid additional shaper latencies, the reserved data rate has to be increased by assuming that in every cycle the maximum occurred amount of data is send. Figure 3.25 shows an example for the shaper configuration. The example shows the send behavior of the packets of the first four communication cycles for both gateways and the different packaging approaches. When looking at gateway 1 and the one-to-one mapping, it can be seen that the maximum number of packets transmitted per cycle is 3. This is the case in the first and the third communication cycle. The average number of packets per cycle is $(3+1+3+0)/4 = 1.75$. 2) If no additional latencies should occur by the AVB shaper, then the data rate allocation calculation has to be based on the maximum number of packets per communication cycle instead of the average number of packets. Note, the calculation based on the average value will also prevent packet loss, but additional queuing latencies can occur.

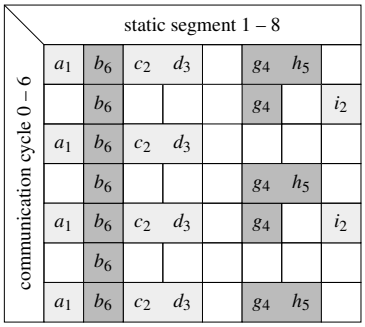
Evaluation Phase During the following evaluation phase, different objectives are of interest. These are: a) The resulting Ethernet data rates for the each gateway, b)



gateway 1	
stream	frame(s)
A	a_1
B	c_2
C	d_3
D	i_2

gateway 2	
stream	frame(s)
E	b_6
F	g_4
G	h_5

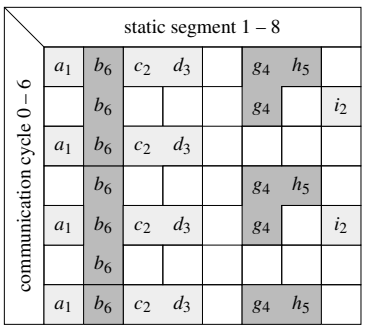
(a)



gateway 1	
stream	frame(s)
A	a_1
B	c_2, d_3
C	i_2

gateway 2	
stream	frame(s)
D	b_6
E	g_4 [, h_5]

(b)



gateway 1	
stream	frame(s)
A	a_1
B	c_2, d_3
C	i_2

gateway 2	
stream	frame(s)
D	$3 \cdot b_6$
E	$2 \cdot g_4$, h_5

(c)

Figure 3.24.: Examples of the different FlexRay to Ethernet AVB mapping approaches. (a) Shows the one-to-one mapping of frames to packets for a FlexRay configuration Matrix **M** and a two gateway system according to Figure 3.19 with corresponding stream configurations (right). Each FlexRay frame is mapped one-to-one to a IEEE 1722 packet. The combination of several frames within the same communication cycle is shown in (b). A mapping of frames over several communication cycles is shown in (c). This approach can be used to replace FlexRay transport protocols.

Figure 3.25.: An example of the send behavior of the packets of the first four communication cycles for both gateways and the different packaging approaches. (a) Shows the initial send behavior of the FlexRay network. (b) Presents the frames related to gateway 1 together with the different mappings to packets obtained from the packaging approaches. (c) Presents the analogue information for gateway 2.

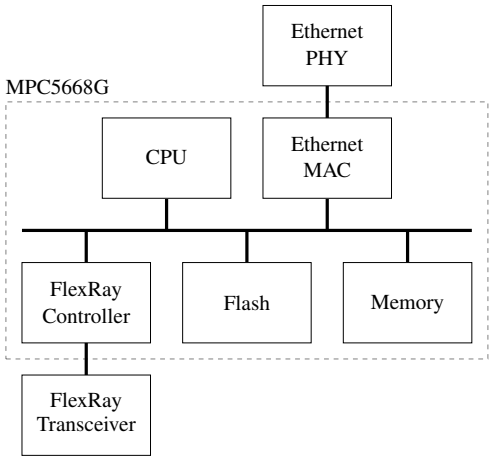


Figure 3.26.: The architecture of the Ethernet/AVB gateway.

the average number of frames per packet, and c) the maximum buffer latencies, for example. The maximum buffer latency denotes the maximum waiting time of a frame within the transmit buffer. A detailed presentation of the results for a real car network configuration is presented in the experimental results section.

3.2.3.2. Test Setup

The prototypical implementation consists of two FlexRay/AVB gateways that are interconnected via an Ethernet AVB switch. To generate and capture traffic from the FlexRay ports of the gateway, a 2-port time synchronized FlexCard from Vector [Vec12b] is used. In addition, port mirroring or sending of broadcast messages allows to trace the traffic through the switch device. With this setup, it is possible to measure different timings like the latency between the two gateways or the jitter of FlexRay frames between the two FlexRay networks. By analyzing the Ethernet packets, the packaging mechanism can be checked and the impact of the jitter on frames caused by the AVB shaper can be visualized. Figure 3.26 shows the architecture of an Ethernet/AVB gateway. In this case, a Freescale MPC5668G 32-bit microcontroller is used to implement the gateway functionality. Since the device is not AVB-capable, the shaper mechanism and the time synchronization have to be realized in software.

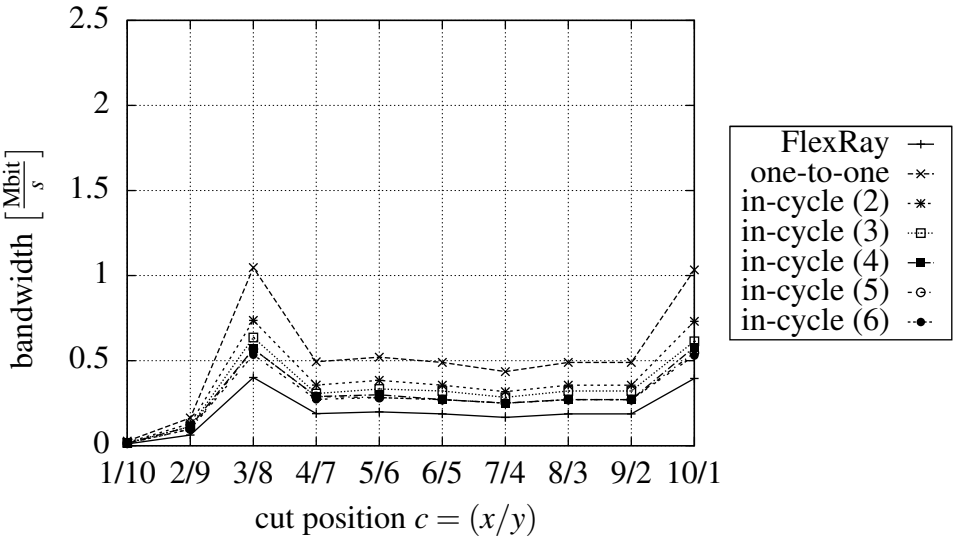
3.2.3.3. Experimental Results

The following experimental results are based on a real FlexRay network car configuration with 11 connected ECUs. The results are presented for different cut positions $c = x/y$ according to Figure 3.19. This means, that x of the 11 nodes reside in the first FlexRay network and the remaining $y = 11 - x$ nodes are placed in the second FlexRay network. The gateways are configured in such a way that they behave like a 'tunnel'. From a functional perspective, the FlexRay nodes should not recognize, that two Ethernet gateways connect the two FlexRay networks. In the following, results on the data rates, the average number of frames per packet, and the maximum buffer latencies are presented.

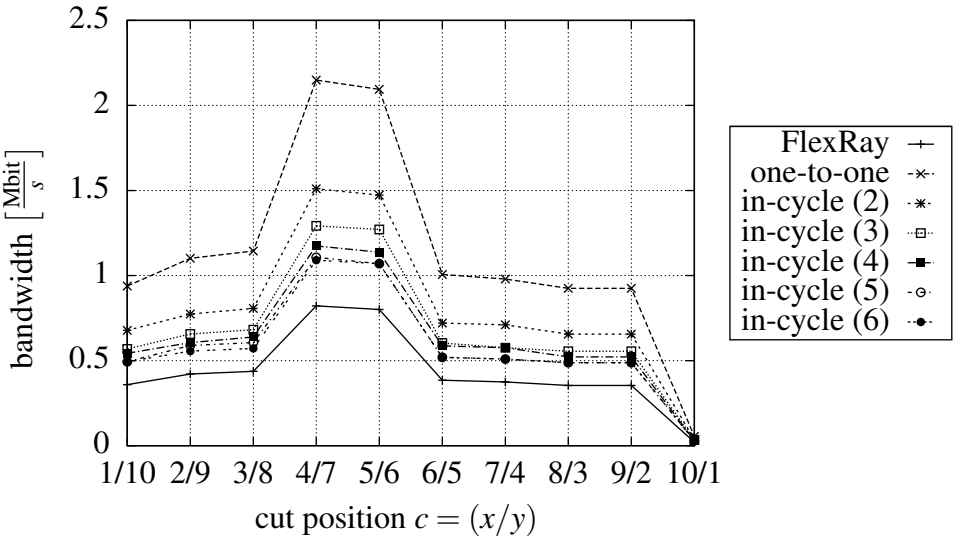
Ethernet Data Rates Figure 3.27 presents the Ethernet data rates for the different packaging strategies compared to the data rates of the FlexRay network. The traffic generated by gateway 1 is shown in Figure 3.27a. The one-to-one approach results in a three times higher data rate than the network load on the FlexRay channel. With buffering, the network load can be halved. For the $c = 3/8$ cut, for example, the FlexRay network has a load of 0.40 Mbit/s and the in-cycle approach resulted in a minimum bandwidth of 0.53 Mbit/s. The network traffic generated by gateway 2 is presented in Figure 3.27b.

Average Frames per Packet The average number of frames per packet for different cut positions are shown Figure 3.28. For the cut positions from $c = 5/6$ to $c = 9/2$ it is interesting to see that the buffer sizes $b_s = 5$ and $b_s = 6$ will lead to the same results. In these cases, that packaging could not be improved by increasing the buffer size b_s.

Maximum Buffer Latency Figure 3.29 presents the maximum buffer time of a frame f within the gateway. For example, when combining three frames f_1, f_2, and f_3 together, the maximum buffer time of this group of frames is the distance of time between the static slot of the last frame f_3 and the static slot of the first frame f_1. The maximum buffer time over all cut positions appeared at cut position $c = 2/9$ and was about 2 ms.



(a)



(b)

Figure 3.27.: The resulting Ethernet data rates for both directions: (a) shows the traffic from gateway 1 to gateway 2 according to Figure 3.19 for different cut positions $c = x/y$. (b) shows the results for the opposite direction.

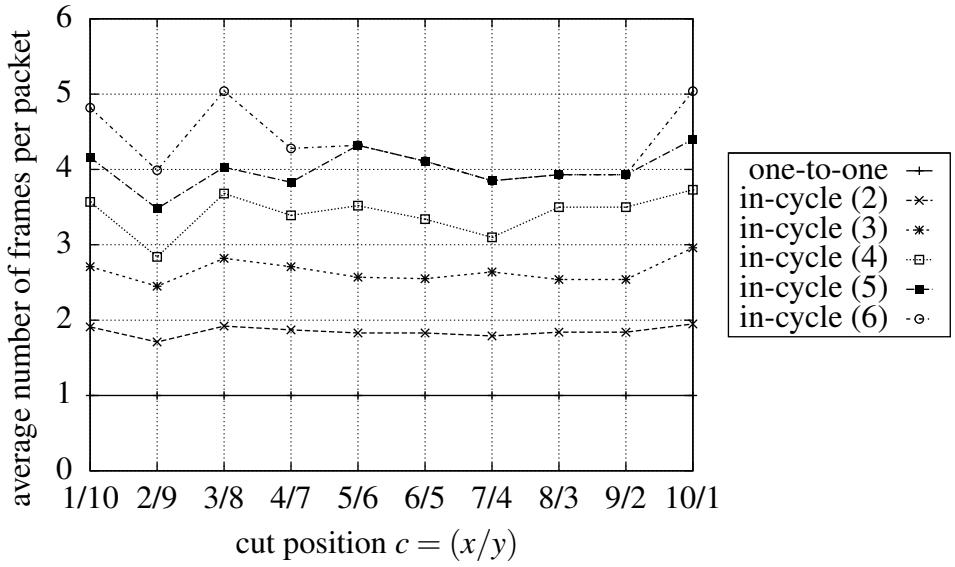


Figure 3.28.: The average number of frames per packet for different cut positions.

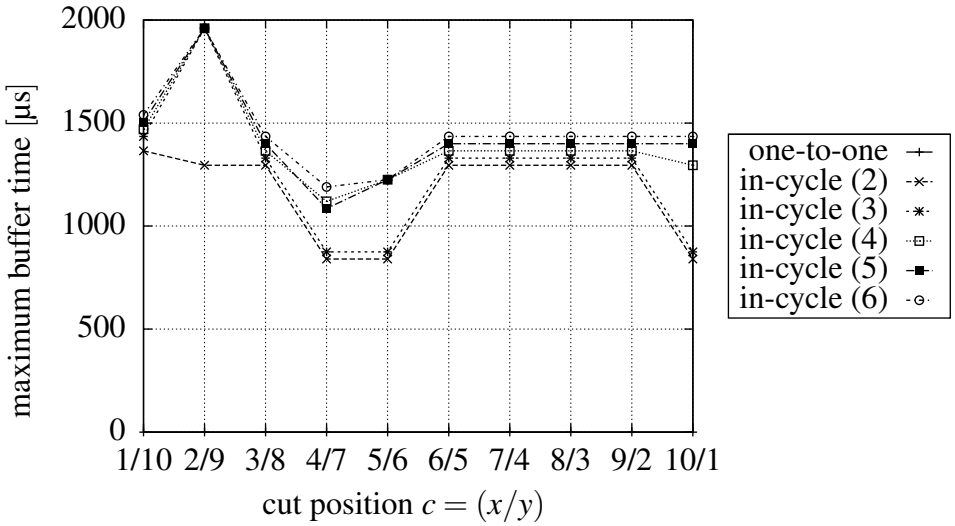


Figure 3.29.: The maximum buffer time of a frame f within the gateway for different cut positions.

3.3. Replacement Approaches

Beside the introduction of Ethernet-capable automotive gateways, the complete replacement of current communication technologies like CAN or FlexRay could be envisioned in the long-term. One big advantage being the avoidance of parallel cabling within same installation spaces. Today, for example, high data rate LVDS connections are assembled to interconnect cameras and ECUs. In addition, a second CAN interconnection is used for control data. In the long-term, Ethernet might replace theses two parallel connections to reduce the cabling effort, to optimize the network topology, and to save costs. This rises the question of whether it is necessary and if yes, how to support the CAN and FlexRay communication protocols via Ethernet. This section introduces two migration concepts for CAN and FlexRay, explains optimization techniques to reduce the number of packets, and presents evaluation results using real communication matrices.

3.3.1. Controller Area Network

When replacing todays CAN communication by Ethernet, different topics have to be considered. The presented migration approach takes into account that CAN communication is based on different send behaviors and supports multi- and broadcast addressing due to the shared physical media. Moreover, an evolutionary-based optimization algorithm for packetization is introduced and compared to a rule-based approach. Extensive evaluations are presented to validate the introduced algorithms.

3.3.1.1. Technology Differences

In the automotive section, a CAN-frame typically includes a *message* (PDU) and the message itself consists of a set of *signals*. Signals represent physical or logical values like current acceleration, light on, or activate seat heating. In the case of Ethernet, IP and higher transport protocols like UDP are used to encapsulate user data into packets. These packets have a header length of 54 byte and a payload length between 18 and 1472 bytes. In contrast, a CAN frame has a header length of 47 bit and a payload length between 0 and 8 bytes which is much smaller compared to Ethernet. The addressing mechanism also differs: CAN uses an 11 bit message identifier which means that every node has to decide by the identifier, if a message

Figure 3.30.: The proposed migration concept.

belongs to it. On the other hand, Ethernet utilizes a 6 byte MAC address which identifies a network interface card. Additional addressing is done by IP and UDP to support logical addresses and port-based addressing. On the physical layer, it is distinguished between shared media when using CAN and point-to-point connections if Ethernet is deployed. In addition, the physical throughputs of Ethernet are at least ten times higher compared to CAN. The *payload/header-ratio* (p/h-ratio) also differs for CAN and Ethernet with the upper protocols IP and UDP. For example, to get the same payload/header-ratio as CAN, a UDP-based solution will need to use up to 75 payload bytes per packet which corresponds to about nine 8-byte CAN-messages.

In summary, three major challenges for migrating the CAN communication to an Ethernet/IP-based communication can be identified:

- *adjustment of send types*
- *address resolution*
- *packaging of messages into packets*

3.3.1.2. Migration Concept

An overview of the considered migration concept is shown in Figure 3.30. It is separated into three basic steps: *preparation*, *optimization*, and *evaluation*. The preparation phase is additionally divided into the three parts *import configuration*, *adjust send types*, and *resolve addressing*.

Configuration Import The configuration import reads a current CAN network configuration from a configuration file and calculates the ECU-dependent message offsets from a network trace with an algorithm presented in [Tra10]. The result is

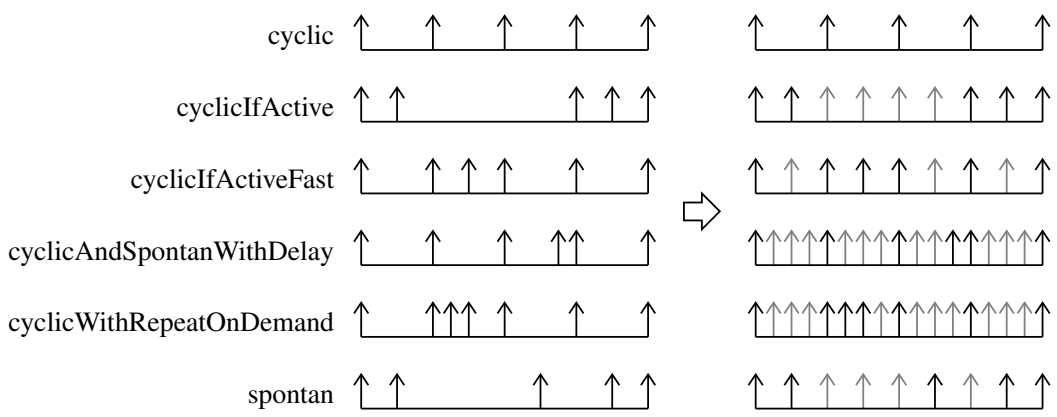


Figure 3.31.: The different CAN send types and their mappings to the cyclic send type.

a set of ECUs E and a set of Messages M. Each message $m \in M$ is defined by the following parameters:

- $s_m \in \{\text{cyclic}, \text{cyclicIfActive}, \text{cyclicIfActiveFast}, \text{cyclicAndSpontanWithDelay}, \text{cyclicWithRepeatOnDemand}, \text{spontaneous}\}$ – the send type of m.
- $l_m \in \{1, ..., 8\}$ – the payload length of m in byte.
- $e^t_m \in E$ – the transmitting ECU of message m.
- $E^r_m \subset E$ – the set of receiving ECUs of message m.

Parameters which depend on the send type of a message are shown in Table 3.6. For example, a cyclic message m consists of a cycle time t^{ct}_m and an offset t^o_m which define the time interval between two messages and the local offset to all other messages of the same ECU.

Send Type Adjustment Current vehicle CAN networks use six different timing approaches which are shown in Figure 3.31 on the left side. These so-called *send types* of messages influence the optimization of the packaging of messages for an Ethernet-based solution, where it is tried to bundle *similar* messages. As a start, we

Table 3.6.: The mapping of the CAN send types and parameters to the cyclic send type for subsequent Ethernet packet conversion.

CAN	**relevant parameters**						**Ethernet**
send type s	cycle time t^{ct}	offset t^{o}	cycle time active t^{cta}	delay time t^{dt}	no. of repetitions t^{nor}	cycle time $t^{ct'}$	offset $t^{o'}$	send type s'
cyclic	t^{ct}	t^{o}				t^{ct}	t^{o}	cyclic
cyclicIfActive		t^{o}	t^{cta}	t^{dt}	t^{nor}	t^{cta}	t^{o}	cyclic
cyclicIfActiveFast	t^{ct}	t^{o}	t^{cta}			t^{cta}	t^{o}	cyclic
cyclicAndSpontanWithDelay	t^{ct}	t^{o}		t^{dt}		t^{ct}	t^{o}	cyclic
cyclicWithRepeatOnDemand	t^{ct}	t^{o}		t^{dt}	t^{nor}	t^{ct}	t^{o}	cyclic
spontaneous		t^{o}		t^{dt}		t^{dt}	t^{o}	cyclic

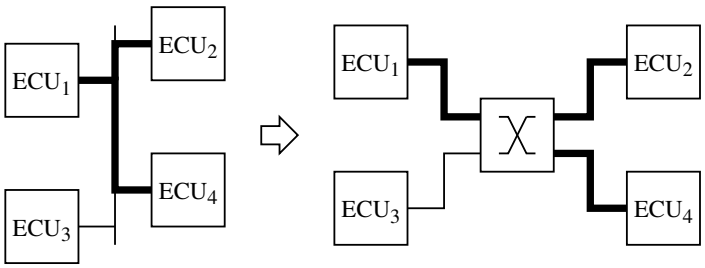


Figure 3.32.: The mapping of the physical-shared topology of CAN to Ethernet point-to-point connections.

decided to map all send types to a cyclic manner to simplify the timing analysis which is quite complex for a mixture of spontaneous and cyclic send types. Furthermore, about 80 % of the messages exchanged in an automotive CAN are already cyclic (see Figure 3.35). Table 3.6 describes the adjustment of the parameters of all deployed send types. For example, the delay time t_m^{dt} of a spontaneous message m which describes the minimal time interval between two messages, is used as the cycle time $t_{m'}^{ct'} = t_m^d$ for the corresponding cyclic message m' in the Ethernet-based solution. As a result, a set of cyclic messages is obtained, each having a cycle time $t^{ct'}$ and an offset $t^{o'}$. The reduction of the send types to just a cyclic send type obviously leads to an overdimensioning of the system requirements because messages are transmitted more often than actually necessary.

Address Resolution The next step in the common process is address resolution which solves the problem of readjusting of the message-based addressing to a node-based address mechanism. Figure 3.32 shows the addressing scheme in a shared media bus topology to a full duplex switched Ethernet topology. Ethernet and IP support *uni-*, *multi-*, and *broadcast* to address one, several, or all end nodes within one subnet. We decided to use unicast addressing due to the following reasons: When using broadcast addressing, every ECU is forced to process all incoming packets to check if the packets belong to it, otherwise the packet is being discarded. In the case of multicast, we have to deal with the fact that most of the switches, in particular cheap ones, do not support multicast. Thus, multicast addresses are handled as broadcast addresses. In addition, multi- and broadcast addressing uses special address ranges which restrict the freedom of address distribution when designing a distributed system. The decision to use unicast leads to an increment of the number of messages

to send, because when a message m in a CAN-based network is received by more than one end node, the Ethernet-based system has to send a packet to each receiver $e^r \in E_m^r$ of this message. The result of this preparation step is a transmitter/receiver configuration matrix containing all messages from one ECU to another within each cell.

Optimization Phase For the optimization of the packetization, Table 3.7 shows three optimization variants. Variant 0 serves as the reference and uses a simple 1:1 transformation strategy which means that every message is encapsulated in its own UDP packet. Variants 1 and 2 use an n:1 transformation strategy to optimize the packaging of CAN frames into UDP packets, i.e., multiple messages are packed into one UDP packet. The assignment is done by an evolutionary approach and a rule-based approach, respectively. UDP is used as a transport protocol due to the reason that a retransmission of cyclic packets does not make sense. Furthermore, we use the UDP port field to identify the receiving processes of a packet p. The result of the optimization step is a set P of packets where each packet $p \in P$ is a set of messages $m \in M$ packed together and containing an Ethernet, IP, and UDP header. The payload length l_p of a packet p in byte is calculated by Equation (3.2). $l(m)$ denotes the length of a message m. In the following, we use the term message for the CAN-based solution and packet to identify the Ethernet-based solution.

$$l(p) := \sum_{\forall m \in p} l(m) \tag{3.2}$$

Table 3.7.: The optimization variants.

variant	0 (reference)	1	2
method	1:1 transformation	n:1 transformation	
	one-to-one mapping	EA-based solution	rule-based solution

Reference – One-to-One Mapping This variant uses a simple one-to-one mapping which means that each CAN frame is encapsulated into exactly one UDP packet, and is used as a reference implementation. This approach has several disadvantages like a huge protocol header overhead due to relatively large protocol headers of UDP-based systems and a large number of packets which are sent via the network. This leads to a higher resource utilization when processing more packets and sending more packets in the network. Equation (3.3) shows the resulting set $P^{1:1}$ of the one-to-one mapping:

$$P^{1:1} = \{p_i | p_i = \{m_i\}\}\ i = 1...|M| \tag{3.3}$$

Variant 1 – EA-based Solution The problem of finding an optimal solution of the packaging of messages into UDP packets is similar to the set cover problem which tries to find an optimal set of subsets covering all elements of a given set of objects. The optimization of such a problem is NP-hard, so a meta-heuristic optimization algorithm to get an approximated solution for the problem is used. For this purpose, the optimization library Opt4J [LGRT11] has been applied which is a Java-based framework implementing an evolutionary algorithm. An integer vector genotype $\vec{g} = \langle n_1, ..., n_{|M|} \rangle$ where $|M|$ is the number of messages m to optimize and n represents the number of the corresponding packets, was chosen to map the memberships of all messages m_i to packets p in the Ethernet/IP-based solution. The parameters of the EA-algorithm were 500 generations with a population of 100, 50 parents, 25 children, and a cross over rate of 0.95.

Variant 2 – Rule-based Solution The following rule-based approach uses a set of rules to find an approximated solution to the optimization problem and is based on a greedy algorithm. The idea is to bundle similar messages into one UDP-packet with the main intention of minimizing the protocol header overhead. Two messages m_1 and m_2 are defined to be similar, when the cycle times of both messages are identical $t^{ct}_{m_1} = t^{ct}_{m_2}$ and the offsets are within a specific range $\left|t^{o}_{m_1} - t^{o}_{m_2}\right| < r$. The approach uses two steps: In the first step, all messages which are similar to each other are combined

into one packet. The first step result is a set P^{rb} of packets p:

$$\begin{aligned} P^{rb} = \{p | \forall m \in M : \exists p \in P^{rb} \wedge m \in p \\ \wedge \forall m_i, m_j \in p : t^{ct}_{m_i} = t^{ct}_{m_j} \wedge \\ \left| t^{o}_{m_i} - t^{o}_{m_j} \right| < r\} \end{aligned} \tag{3.4}$$

In the second step, this approach tries to combine packets p_1 and p_2 with different cycle times $t^{ct}_{p_1}$ and $t^{ct}_{p_2}$ where $t^{ct}_{p_2}$ is a multiple of $t^{ct}_{p_1}$ with $t^{ct}_{p_2} = \alpha \cdot t^{ct}_{p_1}$ and $\alpha \in \mathbb{N}$. With this idea, we can optimize the payload/header-ratio, although the packets with the slower cycle time are sent more often than necessary. Equation (3.5) calculates the total length $l^s(p_1, p_2, \alpha)$ of two packets sent separately and containing all Ethernet, IP, and UDP headers $l^{header} = 54$ byte, fill bytes l^{fill}_p, and messages $m \in p$ in byte. The length of the combined sending $l^c(p_1, p_2)$ in byte is shown in Equation (3.6).

$$\begin{aligned} l^s(p_1, p_2, \alpha) = l^{header} + l^{fill}(p_1) + l_{p_1} + \\ \frac{1}{\alpha} \cdot \left(l^{header} + l^{fill}(p_2) + l_{p_2} \right) \end{aligned} \tag{3.5}$$

$$l^c(p_1, p_2) = l^{header} + l^{fill}(p_1 \cup p_2) + l_{p_1} + l_{p_2} \tag{3.6}$$

The number of fill bytes l^{fill}, which are potentially necessary to obtain the minimum length of an Ethernet frame, is calculated using the following Equation (3.7):

$$l^{fill}(p) = \begin{cases} 18 - l_p & \text{, if } l_p < 18 \\ 0 & \text{, otherwise} \end{cases} \tag{3.7}$$

Figure 3.33 shows the relationship $q(p_1, p_2, \alpha)$ (see Equation (3.8)) of two packets p_1 and p_2 with different cycle times $t^{ct}_{p_2} = \alpha \cdot t^{ct}_{p_1}$. For example, if there are two packets p_1 and p_2 with the payload lengths $l_{p_1} = 16$ byte and $l_{p_2} = 8$ byte and a cycle time factor of 10, it is better to combine these two packets into one new packet $p = p_1 \cup p_2$ with a cycle time of $t^{ct}_p = t^{ct}_{p_1}$, because of getting a better payload/header-ratio (see example point in Figure 3.33).

$$q(p_1, p_2, \alpha) = \frac{l^s(p_1, p_2, \alpha)}{l^c(p_1, p_2)} \tag{3.8}$$

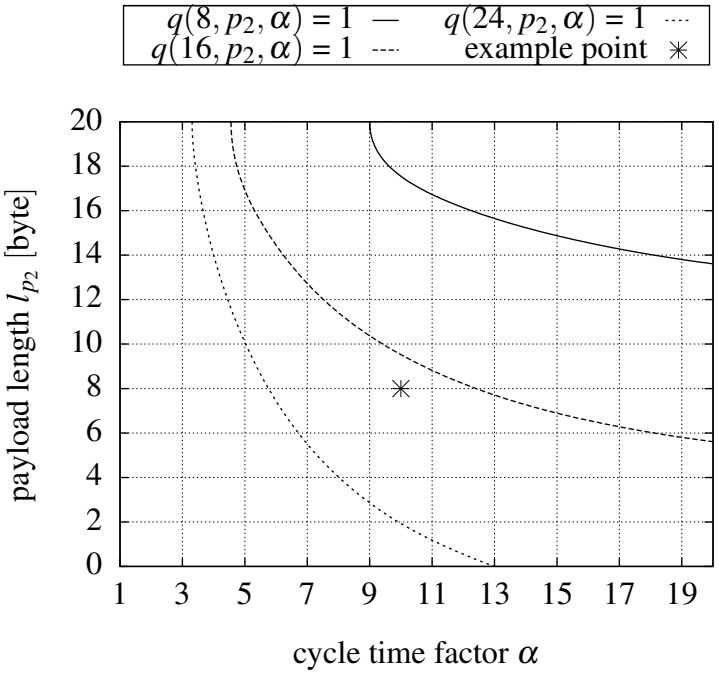


Figure 3.33.: The relationship $q(p_1, p_2, \alpha) = 1$ for different packet lengths of packets p_1 and p_2 and cycle time factors α. The area below the lines indicates a relationship $q > 1$, otherwise the relationship is $q < 1$.

Evaluation Phase This phase evaluates the found solution by using several objectives. The considered objectives are a) the payload/header ratio q^{ph} (see Equation (3.9)), b) the maximum packet sizes, c) the number of packets, or d) the produced offsets when combining several messages with their different offsets into one packet.

$$q^{ph}(p) = \frac{l_p}{l^{header} + l^{fill}(p) + l_p} \tag{3.9}$$

Figure 3.34 shows these objectives for three possible packetization variants using a simple example with five messages m_1 to m_5. For example, solution number one represents the simple one-to-one mapping of these five messages with a resulting payload/header ratio of 10 percent and a low offset rating which means that the offset of the frames packed together are close together. The combined packaging of two messages m_1 and m_2 into one packet p can also produce additional message timing delays, when the two messages have different offsets $t^o_{m_1}$ and $t^o_{m_2}$. In such a case, the earlier message has to be buffered till the other message comes up. This behavior is

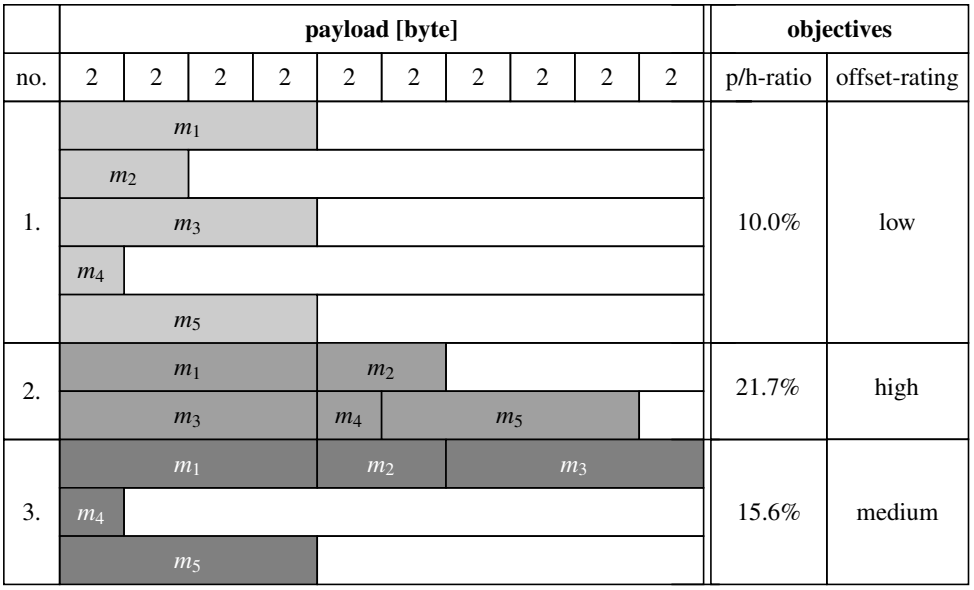


Figure 3.34.: An example of different packetization variants for five messages m_1 to m_5 with the objectives payload/header-ratio and an offset classification.

called *offset-rating*. In the single one-to-one mapping, offset-rating is obviously low.

3.3.1.3. Experimental Results

The migration results are based on current car network configurations. Different BODY-CAN networks from different classes of cars are considered: A compact, middle, and premium car was chosen to show results from a minimum-, a middle-, and a maximum-sized real-world network configuration.

Figure 3.35 shows the results obtained from the preparation phase. The left bar of each car type shows the distribution of the different used send types. The number of different messages varies from 230 to 406 messages and the fraction of the spontaneous messages lies by about 15 percent. The send type adjustment reduces the send types to just a cyclic manner and the address resolution transforms the shared media message-based addressing to a unicast addressing in the Ethernet-based system. This results in a multiplication of the messages by a factor ranging from 2.4 to 2.7 which is represented by the right bar of each car type.

The achieved throughputs for the different optimization variants are shown in Fig-

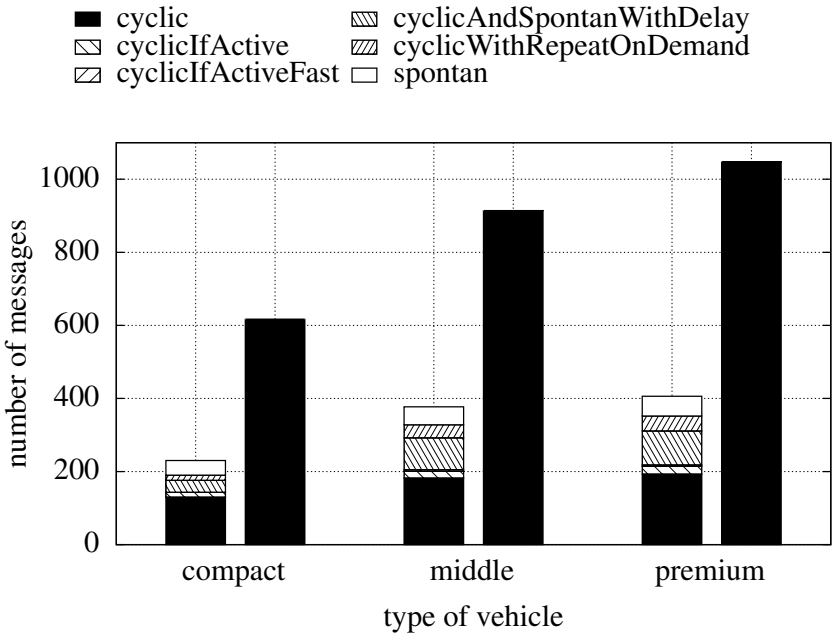


Figure 3.35.: The left bar shows the send type distribution of actual car network configurations and the right bar presents the results from the preparation phase, namely the send type adjustment and the address resolution.

ure 3.36. The presented CAN-based values of the throughputs are calculated after the send type adjustment to include the impact of the send type conversion. It is obvious that the one-to-one mapping leads to the highest throughput results varying from 14 Mbit/s (70x) to 25 Mbit/s (125x). A reduction by half of the values could be realized by using the EA- or rule-based optimization. The throughputs ranges from 8 Mbit/s (40x) to 11 Mbit/s (55x) in the case of an EA-based optimization and between 8 (40x), and 10 Mbit/s (50x) when using the rule-based algorithm. The offset range r was set to 40 % of the respective cycle times. Note that these very high throughputs are worst case throughputs which are mainly caused by the overdimension of the system when adjusting all the send types to cyclic.

Figure 3.37 shows the payload/header ratio for the CAN-based solution and the different optimization variants. The CAN-based values are again calculated after the send type conversion and are over five times higher compared to the one-to-one mapping. In the case of the EA- or rule-based optimization, the payload/header-ratio values range from 16 % to 22 %.

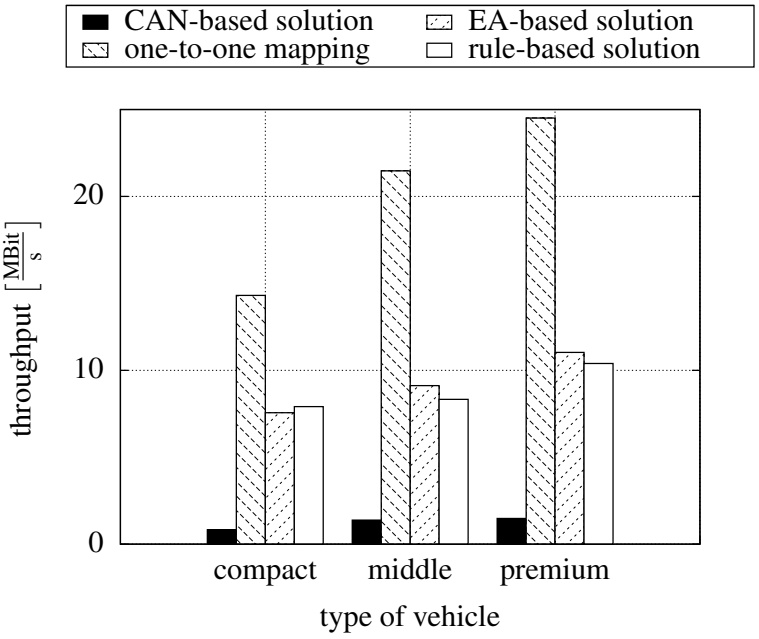


Figure 3.36.: The required Ethernet throughput of the different optimization variants compared to the CAN-based network design.

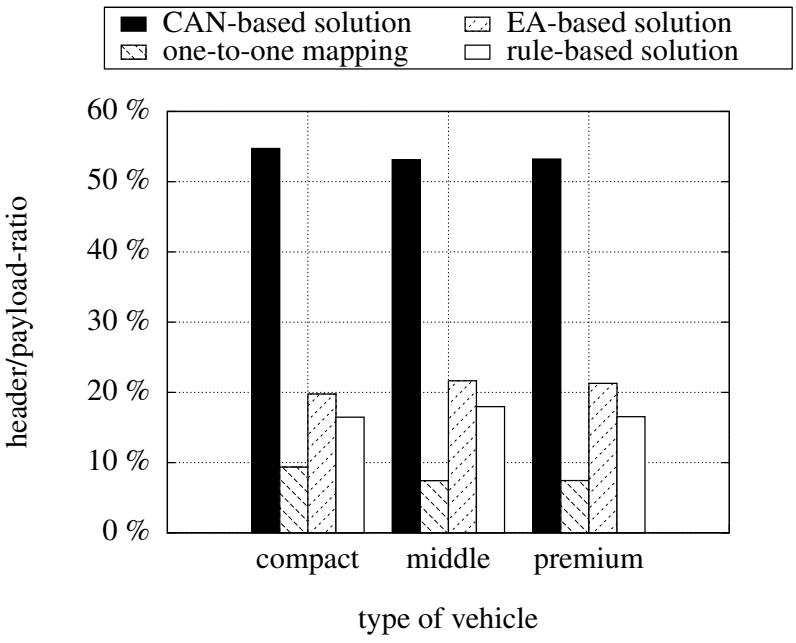


Figure 3.37.: The payload/header ratio for the different optimization variants compared to the CAN-based network design.

Note, that these results are worst case results by mapping all send types to cyclic send types.

3.3.2. FlexRay

As in the case of CAN, different problems have to be considered when encapsulating FlexRay communication by Ethernet. The possibility of using uni-, multi-, and broadcast addressing for FlexRay has to be mapped to Ethernet and the packaging of FlexRay frames into Ethernet packets has to be optimized. The following concept is derived from the FlexRay/AVB gateway concept and evaluated by setting up a prototypical hardware implementation which is connected to an Ethernet packet generator and trace tool to measure the upcoming data rates, latencies, and jitters.

3.3.2.1. Technology Differences

FlexRay also uses the principle of including a message into a FlexRay frame and the message itself consists of a number of signals which represents logical or physical values. In the case of Ethernet, transport protocols like IEEE 1722 can be used to transport FlexRay information. These packets have a header length of 42 byte and a payload length between 22 and 1476 bytes. In contrast, a FlexRay frame has a header length of 8 byte and a theoretical payload length between 0 and 254 bytes. As described in the gateway section, the used payload length in the automotive sector is much smaller than the 254 bytes. By assuming a payload length of 40 bytes, up to 30 FlexRay frames can be embedded within just one IEEE 1722 packet. The addressing also differs. A FlexRay frame can be identified by its cycle and slot position. Instead Ethernet uses point-to-point connections and transmitter/receiver addresses.

3.3.2.2. Migration Concept

The migration concept is derived from the transformation concept of the FlexRay/AVB gateway and is shown in Figure 3.38. It is separated into the four steps *Import Configuration*, *Address Resolution*, *Optimization of Packetization*, and *Evaluate Solutions* and are presented in the following.

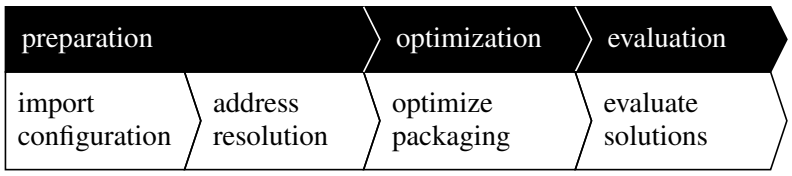


Figure 3.38.: The proposed migration concept.

Configuration Import The configuration import is almost the same as in the gateway concept. Here, a subset of ECUs can be defined, to reduce the number of ECUs. This is needed for the implementation due to only having an eight port packet generator and trace tool.

Address Resolution The mapping of uni-, multi-, and broadcast addressing is realized by the introduction of *virtual networks* (VLANs). For each communication pair consisting of a transmitter and one or more receivers, a virtual network with its own identifier is created. Figure 3.39 shows an example with three communications pairs which are mapped to three different virtual networks. For example, the frames a_1, c_2, d_3, and i_2 are sent by ECU_1 to ECU_2 and ECU_3 and are mapped to virtual network 1. The frames are then sent within a tagged Ethernet frame containing the VLAN identifier 1 and using the broadcast address as the destination address. In addition, the switch has to be configured with the VLAN table which describes the port to VLAN mappings. Table 3.8 shows an example of such a mapping table. In this case, the ports 1, 2, and 5 are mapped to VLAN 1 for example. This ensures that an incoming tagged frame is only forwarded to those receivers which are mapped to the corresponding virtual network.

Optimization of Packetization For the optimization of packetization, the same approach as in the presented gateway concept is used and applied. It is distinguished between a one-to-one mapping which maps every FlexRay frame into one IEEE 1722 packet and the in-cycle and multiple cycle approaches which try to combine several frames into one IEEE 1722 packet.

Evaluate Solutions During the evaluation phase, a) the data rate, b) latency, and c) jitter of a prototypical implementation are evaluated. The results are presented in the following.

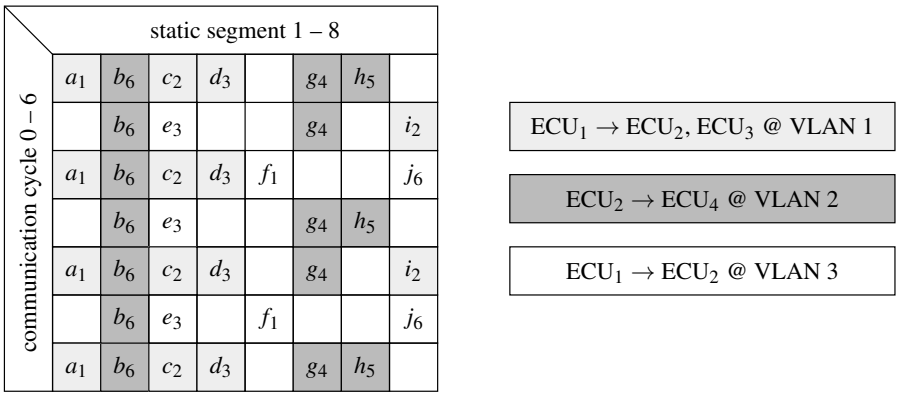


Figure 3.39.: The mapping of the communication pairs consisting of a transmitter and one or more receivers to their virtual networks.

3.3.2.3. Experimental Results

To evaluate the proposed migration concept, a real FlexRay network configuration was taken and the data rate, latencies, and jitters of the corresponding Ethernet-based systems were measured. Due to only having an eight port packet generator, a subset of ECUs was imported to configure the packet generator and trace tool. In addition, the switch was programmed with the calculated mapping table to ensure, that each incoming packet is only forwarded to the necessary outgoing ports. Figure 3.40 shows the topology of the prototypical implementation. For the generation of packets, the tracing of packets, and the measurements Spirent TestCenter was used. Therefore, the send behavior of each ECU is modeled within Spirent TestCenter and each port which corresponds to exactly one ECU is directly connected to the switch. The mea-

Table 3.8.: An example of a switch mapping table. It shows the mapping of each port to its participating VLANs. A T denotes, that the port participates in the virtual network.

Port	**1**	**2**	**3**	**4**	**5**	**6**	...	*n*
VLAN 1	T	T			T			
VLAN 2		T	T					
VLAN 3		T	T	T				
...								
VLAN *m*								

sured results are presented in the following.

Data Rates Figure 3.41 shows the measured data rates of the Ethernet-based system for the different replacement approaches compared to the initial FlexRay data rate. The data rate of the FlexRay subset network consisting of eight ECUs is about 0.9 Mbit/s. By using the one-to-one approach, the total transmitted data rate for all ECUs is about 2.3 Mbit/s which is factor of 2.5 compared to FlexRay. The higher receive data rate of about 1 Mbit/s is caused by multiple receivers for different messages. In this case, the switch forwarded incoming packets to multiple outgoing ports. The in-cycle approach helped to half the transmit data rate by combining multiple frames into one IEEE 1722 packet. Further improvements has been achieved by using the multiple-cycle approach. In this case, typical FlexRay transport protocols are replaced by combining theses frames into single packets.

Latencies and Jitters The end-to-end latency is calculated by the difference between the send and receive time of Spirent TestCenter and thus includes also the buffer time within the switch. Figure 3.42 shows the minimum, average, and maximum measured latencies over all sent and received packets for the different approaches. The minimum latency is the same for all approaches and mainly contains the store-and-forward time of the switch for one packet without any additional queuing delays. The maximum delay occurs during startup when every node starts sending. All ports of Spirent TestCenter are synchronized during the measurements. Figure 3.43 presents the measured jitter values for the different approaches. It is shown, that the average jitter times are in the range of a few microseconds. The maximum jitters occur again during the startup time when all nodes start sending.

In summary, it could be shown that the throughput overhead is relatively low. In case of the multiple-cycle approach, where even transport protocols are resolved, the throughput is almost identical. The relatively high maximum latencies and jitters caused during the start up phase would also be lower when having real systems with varying start up times. Thus, replacing FlexRay by Ethernet AVB do not cause any problems when looking at throughput, latency, and jitter for this given network configurations.

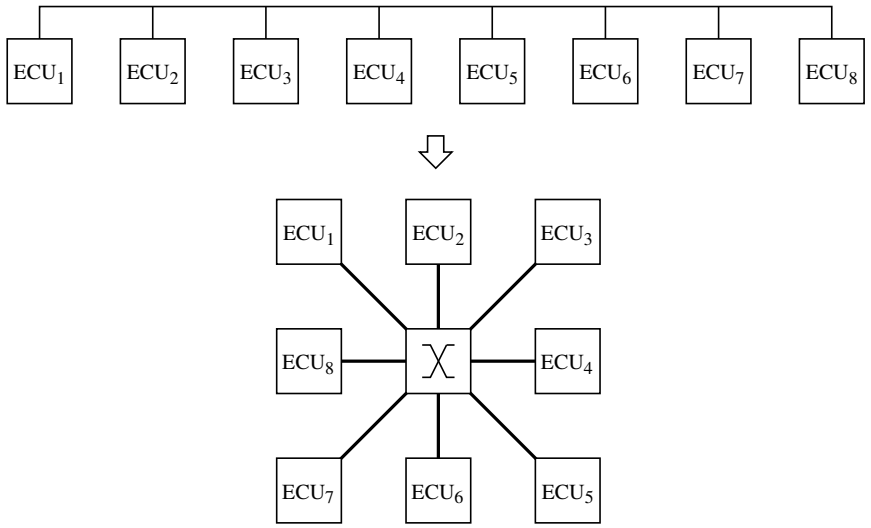


Figure 3.40.: The migration concept. In this case a whole FlexRay network will be replaced by an Ethernet network. Each ECU is therefore connected to a central switch.

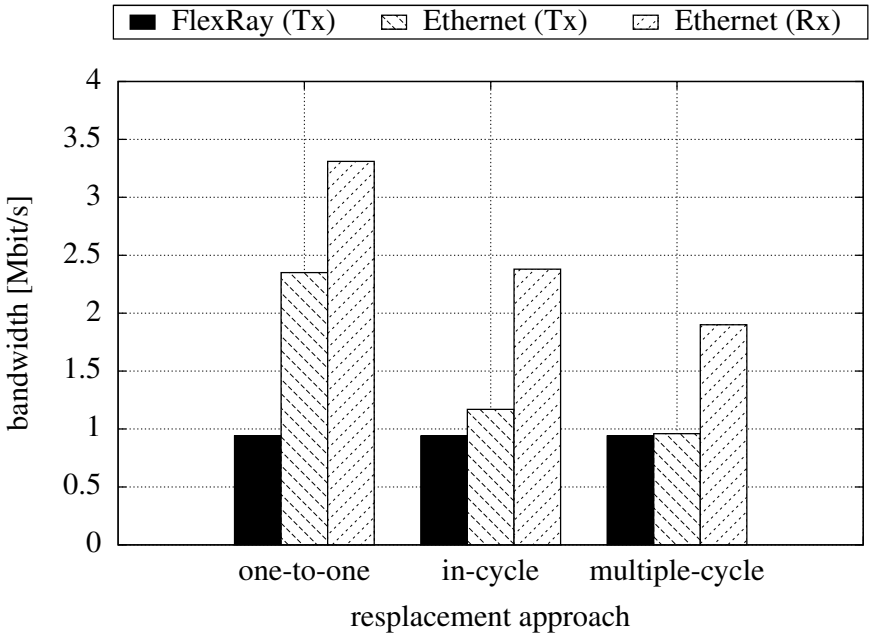


Figure 3.41.: The resulting Ethernet data rates for the different replacement approaches compared to the initial FlexRay data rate. Due to multiple receivers, the receiver data rate is higher.

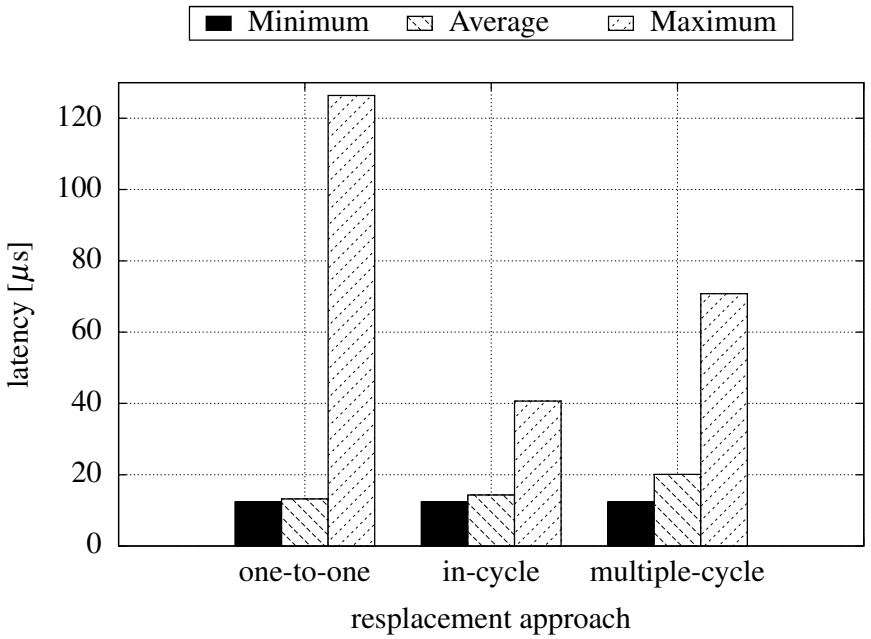


Figure 3.42.: The minimum, average, and maximum measured latencies for the different replacement approaches.

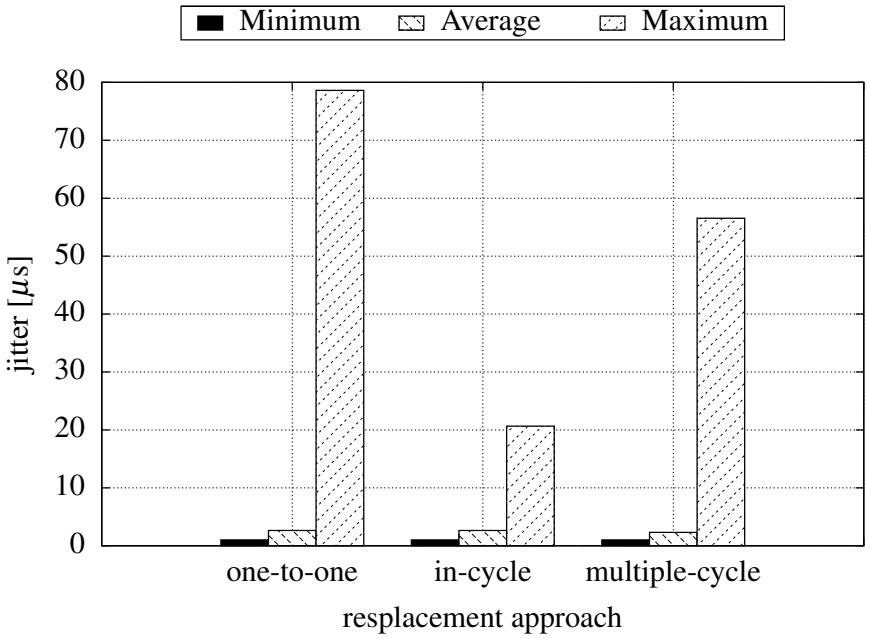


Figure 3.43.: The minimum, average, and maximum measured jitters for the different replacement approaches.

3.4. Summary

Due to the fact that nearly every car uses a common network infrastructure the question of how to integrate a new communication technology within an existing network will arise. This chapter presented gateway strategies to connect Ethernet/IP to the existing E/E communication technologies CAN and FlexRay. Prototypical implementations confirmed the realizability and detailed timing evaluations were provided. As a long-term scenario, the replacement of existing networks was also considered. Detailed migration concepts were introduced for the replacement of CAN and FlexRay. The effects of the different replacement approaches were analyzed by real network configurations of current vehicles and the results were presented.

In the gateway area, a CAN/Ethernet gateway concept was presented, implemented, and finally tested within a vehicle by tunneling a BODY-CAN network via Ethernet/IP. The evaluations were based on stress tests by the definition of CAN input traces with different data rates and real network configurations based on a BODY-CAN network. The average end-to-end latency ranges from 500 µs in case of the one-to-one mapping without buffering up to 4500 µs in case of the buffered approach for CAN resource utilizations of up to 80 %. The average throughput without buffering is about 1.2 Mbit/s when tunneling a 40 % utilized 500 kbit/s CAN network. Buffering reduced the Ethernet data rate down to 320 kbit/s in the case of the urgency approach together with a buffer timeout b_t of 10 ms. The maximum resource consumption of the gateway processor is about 27 %. In addition, a FlexRay/AVB gateway concept was introduced to forward data of the static segment of FlexRay to Ethernet AVB and vice versa. The transformation concept covers in-cycle and multiple-cycle buffer approaches to replace typical transport protocols for FlexRay. The evalution results are based on a real FlexRay network configuration which is splitted into two subnetworks. The maximum resulting Ethernet AVB throuphput is about 2.2 Mbit/s for a 800 kbit/s utilized FlexRay network and the maximum frame buffer time for this case is about 1.2 ms. In summary, it is shown that is possible to connect CAN or FlexRay with Ethernet and AVB and that the packaging overhead is relatively low compared to the different header sizes.

A migration concept for long-term replacement of CAN or FlexRay was presented in the second part of this chapter. First, a CAN over Ethernet concept was introduced and evaluated. The packaging optimization is based on a reference one-to-one

mapping, an EA-based approach, and a rule-based approach. The results show worst case data-rates by assuming asynchronous messages as cyclic messages with defined debounce times. For a 40 % utilized 500 kbit/s CAN network, a worst case Ethernet data-rate of about 24 Mbit/s using a one-to-one mapping approach was achieved. The EA-based und rule-based approches are able to half the data rates in the Ethernet network and led to a header overhead of about 20 %. In the case of FlexRay, the FlexRay gateway concept was adopted and the different connections were mapped to different VLANs. A test setup using the Spirent TestCenter showed that a transmit data rate of about 900 kbit/s led to an Ethernet AVB transmit data rate of about 2.4 Mbit/s. Through buffering, the send data rate could be reduced down to about 950 kbit/s which is close to the FlexRay data rate. Furthermore, the average latency was comparable to the latency caused by the FlexRay network. In summary, it could be shown that it is possible to replace a CAN or FlexRay network by a Fast Ethernet network.

In summary, when introducing Ethernet/IP in future vehicles, a redesign of the communication matrices should be considered to make use of the advantages of Ethernet/IP (e.g. large payloads of up to 1500 bytes). Legacy communication from or to existing communication systems like CAN and FlexRay could then be adjusted by a application gateway which can perform more complex adjustments than just doing internal memory movements as in the case of todays typical gateways.

4. Simulation and Pre-production Testing of Ethernet/IP-based E/E-Architectures

In order to minimize the guarantee and goodwill costs of a vehicle, testing and evaluation are fundamental methods to reduce the failure probability of systems within the vehicle. Figure 4.1 shows the well-known *1-10-100 rule* with respect to the automotive section. It is shown that the error cost factor increases exponentially during the product life cycle process. An error during the design and development phase has a factor of one to fix. During the preparation phase, an error costs typically ten times more than in the development phase. After *Start of Production* (SoP), the error costs are 100 times higher than before. Finally, during the utilization phase when the car is used by the customer, a failure will generate error costs which are up to 1000 times higher than the costs when fixing the same error during the development phase. That is why error prevention and error finding in early stages of the development process are fundamental points when developing a new vehicle [Pfe01].

Automotive network technologies are basic infrastructure elements and must be extremely reliable to guarantee the functionality of the whole vehicle. A malfunction of the powertrain network, for example, would lead to a shutdown of the engine of the car. Testing and evaluation of such network technologies can be divided either by using the OSI-layer model as presented in Figure 2.2 where a method is designed to check the layer 1 and 2 robustness, for example, or it can be separated into the different stages of the product life cycle. The concepts presented in this section focus on the early stage of product life cycle – the development process. It is distinguished between testing and evaluation of systems in the concept phase where no hardware or software is available and systems which are partly or completely available, for

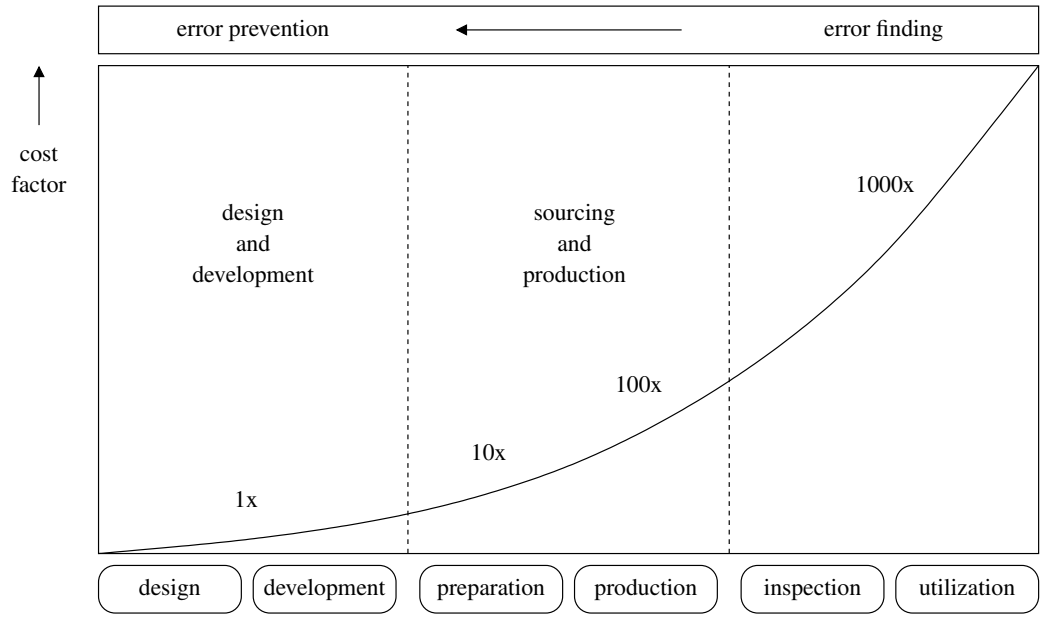


Figure 4.1.: The error cost factor evolution during the product life cycle process [Pfe01].

example, when receiving the first prototypes by the suppliers.

In this chapter, two approaches are proposed to simulate and test Ethernet/IP-based embedded systems:

Simulation Environment A simulation environment can be used to examine the technical feasibility of an application concept, to support the hardware and software architect during the design of a system, or to determine the resource consumptions of the different components. The presented concept is based on a *Discrete Event Simulator* (DES) which is extended by an Ethernet PHY, an Ethernet MAC, an Ethernet switch, and a task schedule model. All components are fully configurable to fit the automotive requirements. The configuration parameters of the different components are provided by data sheets or by measuring available devices. With this approach, it is possible to model Ethernet-based networks and end nodes to get a first impression (e.g., task scheduler utilization and memory usage) of the behavior of the modeled Ethernet-based system. For example, the simulator allows to calculate the throughputs on the different links within the system. This is a very helpful information for the hardware architect when designing the network topology of a new vehicle. In the following, we will focus on the evaluation of an automotive case study based

on a multi-camera system which is presented and evaluated by the Ethernet-capable simulator and the results are compared with an analytical approach.

Transparent Ethernet Switch This concept is designed to test systems which are partly or completely available. Therefore, a special Ethernet-Test-Switch design will be presented which works almost completely transparent to the application, but allowing the complete tracking or insertion of packets. In addition, stress tests can be executed or components which are not yet available in hardware can be simulated by attaching the Ethernet/IP simulator from above to the Ethernet-Test-Switch. This enables the manipulation of packets routed through the switch by the simulator. The testing of partly available systems by using simulation tools for devices which are not available is called *restbus simulation*.

4.1. Fundamentals

The following section gives an introduction to discrete event simulation and compares the differences between physically-shared medium technologies and switched technologies in the context of testing.

4.1.1. Introduction to Discrete Event Simulation

Discrete event simulation is a strategy to simulate the operation of a system by using events. An event $e \in E$ can occur at a time t and can change the state S of a system. For example, when an Ethernet communication controller receives a $\langle\langle$new packet p arrived for sending$\rangle\rangle$ event, the communication controller will either create an internal $\langle\langle$add packet p to internal transmit buffer $b_t\rangle\rangle$ event and queues the packet or it will add the Ethernet frame header and sends the packet to the PHY by creating a $\langle\langle$new frame $f(p)$ for sending$\rangle\rangle$ event after a defined duration d_s which contains the packet p together with the Ethernet frame header. Every duration can be represented by a distribution function. The example illustrates that the discrete event simulation enables the possibility to build modules like an Ethernet controller or PHY. Each module has a defined interface $I = E_r \cup E_t$ for all received E_r and transmitted E_t events and an internal state machine represented by a Petri net [Pet62] P, for example, which models the functionality of the component. When using a Petri net

to model the behavior of the module, incoming events can be represented as tokens of input places of a transition, for example.

Beside the creation of the different modules with their interfaces and state machines, Figure 4.2 shows the general event scheduling scheme when using computer simulation. All events are stored in one central 'Schedule Event List'. After starting a simulation, a module typically creates an event which triggers subsequent events, coordinated by the different state machines of the modules. The simulation ends 1.) after a defined time interval t_i, 2.) a defined condition like 'buffer overflow', 3.) or when the schedule event list is empty. The procedure of processing an event within the schedule event list is as follows:

1. Remove first entry (e_1, t_1) from schedule event list.
2. Update time to the new event time t_1.
3. Update the system state S consisting of the different module states x.
4. Remove all infeasible events in the new state.
5. Insert new events to the schedule event list ordered by their schedule time.

4.1.2. Differences in Physical Shared and Switched Technologies

This section discusses the differences in terms of testing between today's physical shared medium technologies and switched Ethernet networks.

Physical Shared Medium Technologies Todays established communication technologies in vehicles are based on a physically shared medium, a so-called bus, as shown in Figure 4.3(a). In such a system, each communication transceiver is connected to the same physical wire. Regarding testing, this is an important ability since every frame on the bus is visible to every participant. As a result, it is possible to add a test device that can monitor the whole bus traffic and is able to send frames to any bus participants. It is also possible to simulate the network behavior of complete devices through a simulation environment within the test device.

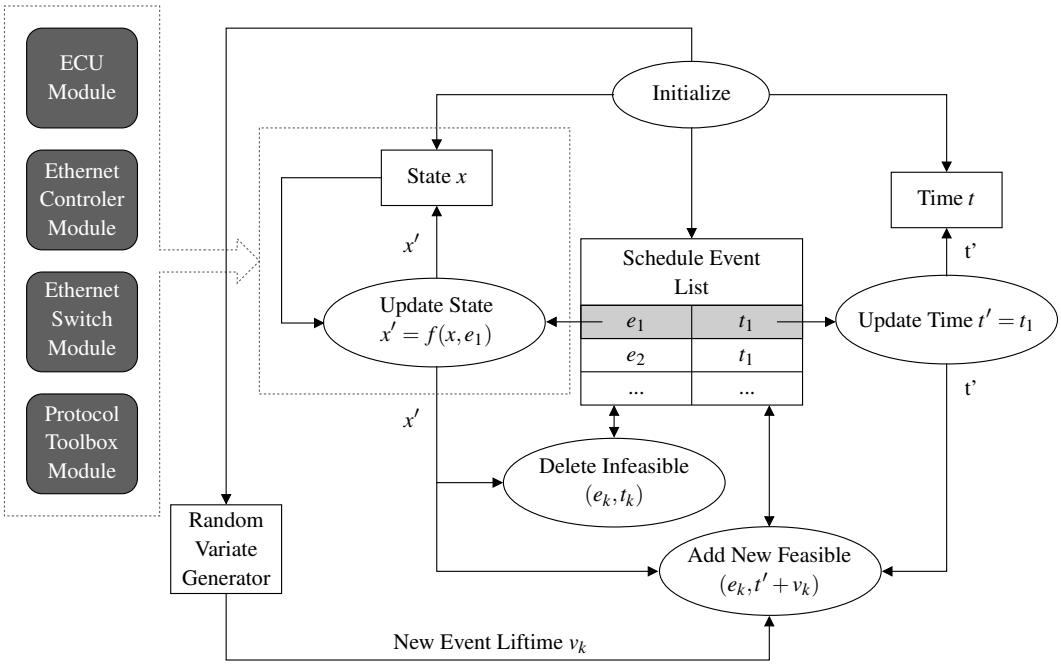


Figure 4.2.: The general event scheduling scheme in computer simulation together with the different Ethernet modules which are located in the ⟨⟨Update State⟩⟩ section [CL08].

Switched Ethernet Networks In switched Ethernet networks, a switch is used to interconnect several endpoints as shown in Figure 4.3(b). A similar approach of just adding an additional test device to a port of the switch is not appropriate, since switches use an intelligent forwarding approach to avoid broadcasting of packets. This means that switched packets are not forced to be forwarded to the test device. The forwarding mechanism works as follows: the destination MAC-address of each incoming packet will be compared with the entries of an internal lookup table. If an entry for a certain MAC-address exists, the packet will only be forwarded to the deposited port. If no entry exists, the packet is broadcasted to all ports of the switch. Additionally, every source MAC-address of incoming packets is added with the port number of the received frame is added to the internal lookup table. When disconnecting a device from a port of the switch, the MAC-entries for this port are discarded. In order to handle this problem, a special approach is needed which provides an insight into the switch to monitor all packets which are processed by the switch fabric. This feature makes monitoring of traffic much more difficult than for shared communica-

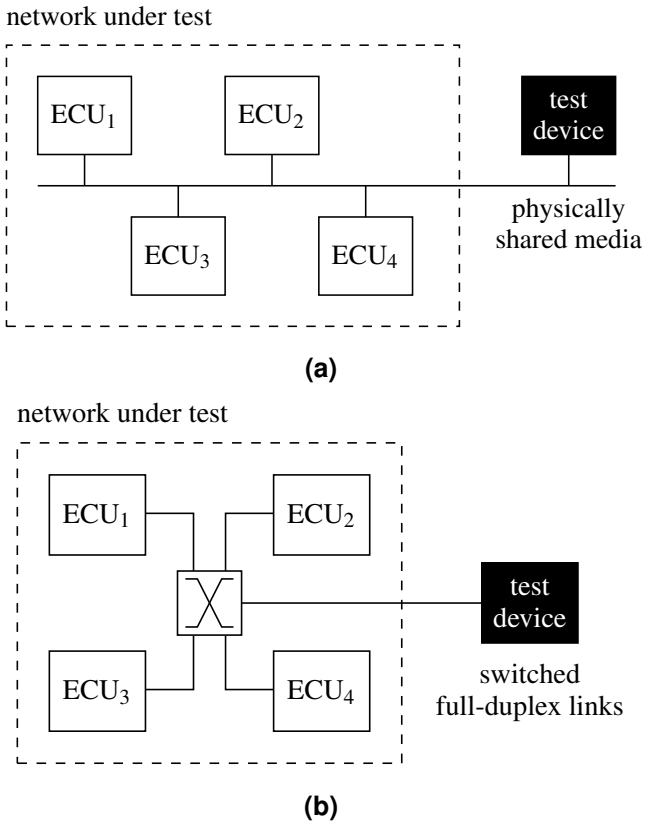


Figure 4.3.: Differences between physical shared and switched network topologies: (a) Shows a typical test setup for physical shared medium communication technologies like LIN, CAN, or FlexRay. (b) Demonstrates an intuitive test setup for switched Ethernet networks.

tion media.

4.2. Related Work

The related work can be divided into three main groups: network simulation, packet monitoring, and packet generation. In the network simulation area, there exist several software tools like [HH09] or [Vec11] to simulate Ethernet-based networks. The latter does also supports external interactions from an Ethernet interface. The main disadvantage is the lack of support for Ethernet embedded systems including a) software network stack simulation and b) the consideration of different software process schedulers. When also wanting to connect a software simulator to existing devices to

enable restbus simulation, todays tools do only provide one physical network interface. One feasible solution to receive traffic from several devices within one simulation tool could be the usage of a switch together with port mirroring [ZM07]. This technique copies traffic from one or more ports to an additional port – the so-called mirror port – where the tool can be connected. Disadvantages of this approach are the limited throughput of the mirror port and sequential transmission of parallel arriving packets over the mirror port. For packet monitoring, there exist several different concepts. In [Net11], an Ethernet point-to-point connection is physically splitted within a special device to enable the possibility of adding a third monitoring device. A similar approach is used in [OV05] where a computer with a special software tool and two network interfaces acts as a man-in-the-middle. These two approaches do only work for point-to-point connections. When having multiple interfaces connected via a switch, the port-mirroring approach can be used again. Two similar commercial solutions for packet generation are presented in [Spi11] and [Ixi11]. The tools support the generation of statically predefined packets up to full data-rate. The main task of these products is the performance analysis of Ethernet-based networks.

4.3. Simulation-based Evaluation of Switched Networks (Case Study)

In the following, a simulation-based approach is introduced to simulate the behavior and timing of Ethernet-based networks. Therfore, a network simulator is adapted with Ethernet-specific components and automotive-specific parameters are used to simulate automotive embedded Ethernet-based systems. The concept is validated by the evaluation of a multi-camera system use case based on Ethernet/IP. In addition, the same use case is evaluated using an analytical approach to obtain the gap between the simulated worst case delay and the worst case calculated by the analytical method.

4.3.1. Simulation Concept

Figure 4.4 shows an example of an automotive-typical task chain which consists of three tasks t_1, t_2, and t_3. In this example, task t_1 communicates with task t_3 via Ethernet/IP through the switch and task t_2. The *Stack* component represents the IP

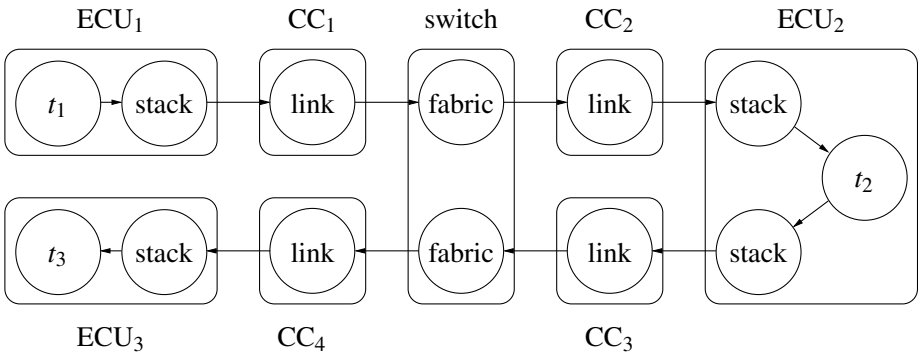


Figure 4.4.: An example of a task chain which consists of three tasks t_1, t_2, and t_3 communicating via Ethernet/IP. In order to simulate such a network, the Ethernet *Communication Controller* (CC) and the Ethernet *switch* have to be modeled.

and higher layer transport protocols. Beside the software tasks t_i and the software network stack, the Ethernet *Communication Controller* (CC) and the Ethernet *Switch* have to be modeled to simulate an Ethernet-based System.

The network stack component is responsible for adding the different protocol headers including different protocols like TCP which supports handshakes and automatic retransmissions of packets in the case of packet loss during transmission. A *First In First Out* (FIFO) buffer within the communication controller coordinates the transmission of packets. Each buffer has a predefined size b_s in bytes and some implementations share the buffer memory for incoming and outgoing packets. The switch receives packets and forwards them to the known outgoing ports. There exist different methods to forward packets from incoming to outgoing ports and the switch can automatically learn the content of its lookup table. At the receiving side, an incoming packet is processed by the software network stack. Therefore, the protocol headers are analyzed and removed. The resulting application data is forwarded to the receiving task. In the following, different concepts for modeling the component behavior and timing are presented.

Switch Modeling Ethernet switches are used to interconnect different ECUs and to build larger network topologies. Their main task is the forwarding of packets from incoming to outgoing ports. However, the forwarding method is not part of the standard and therefore manufacturer-dependent. In order to simulate an Ethernet switch,

a derived forwarding method has to be assumed. In general, there are two types of switching mechanisms. The backplane implementation which uses an internal high speed bus to interconnect the different ports and the so-called crossbar architecture which enables dedicated point-to-point connections between the different ports of the switch. In the case of a backplane, the simple *Round Robin* (RR) method is used for backplane arbitration. The time slot t_{switch} denotes the duration which is needed to forward a complete Ethernet packet by the backplane. The *Maximum Transmission Unit* (MTU) l_{MTU} is typically 1518 bytes for untagged Ethernet packets. For a given backplane speed of b_{switch}, the duration for the transmission of one packet t_{switch} can be calculated by Equation (4.1).

$$t_{\text{switch}} = \frac{l_{\text{MTU}}}{b_{\text{switch}}} \tag{4.1}$$

Beside the forwarding mechanism, the lookup table is another important part of a switch. It can be either statically predefined or dynamically filled. Dynamic tables learn the addresses of the different nodes connected to a switch port by analyzing the transmit address of incoming packets. Whenever a packet arrives at a port, the switch adds the transmit MAC-address to its internal lookup table. Packets are then forwarded to the outgoing port, where the destination address is assigned. Additional functionalities are the support of multicast addresses, tagged frames, and virtual networks.

Communication Controller Modeling An Ethernet communication controller has to support different speeds ranging from 10 Mbit/s to 10 Gbit/s and the full-duplex mode allows the parallel transmission and receivement of packets. In order to simulate this behavior, two resources are instantiated, one for sending packets and one for receiving packets. In addition, each controller contains a buffer with a predefined buffer size b_s to model the loss of packets due to a full buffer. It is also possible to set the buffer size to infinity in order to calculate the maximum buffer sizes for each communication controller. Hardware MAC address filters and the support of tagged Ethernet frames can also be modeled to reduce the number of software interrupts.

Packet Modeling An untagged Ethernet packet has a maximum length of 1538 bytes including the preamble (8 bytes) and the *Inter Frame Gap* (IFG) (12 bytes)

[IEE12b]. The effective transmission length l_{MAC} including the Ethernet headers, preambles, and frame gaps for a given payload length l_p can be calculated by Equation (4.2).

$$l_{\mathrm{MAC}}(n) = 38\left\lceil \tfrac{n}{1500} \right\rceil + n + \max\{0\ ;\ 46 - (n \bmod 1500)\} \tag{4.2}$$

Since every Ethernet packet has a minimum length of 64 bytes, additional fill bytes are used to transport payload lengths smaller than 46 bytes. Payloads larger than 1500 bytes are separated into multiple frames by higher layer protocols. In addition, higher layer protocols typically have their own protocol headers like IP which has a minimal header length of 20 bytes or UDP with a length of 8 bytes. The effective transmission length l_{UDP} of a payload length of l_p bytes for an Ethernet packet containing IP and UDP protocol headers can then be calculated by Equation (4.3).

$$l_{\mathrm{UDP}}(n) = 58\left\lceil \tfrac{n+8}{1472} \right\rceil + n + 8 + \max\{0\ ;\ 26 - ((n+8) \bmod 1472)\} \tag{4.3}$$

4.3.2. Multi-Camera Use Case

For the evaluation of the implemented simulation components, a case study based on an Ethernet multi-camera system is used. The Ethernet components are integrated into a discrete event simulator [HH09] which simulates the use case described in the following. In addition, exactly the same use case and modeling is used by an analytical approach to compare the results of the simulation and the analytical approach.

System View The considered multi-camera system has the following functional principle: multiple cameras cam_i capture defined surroundings of the vehicle. These pictures are aggregated within a central ECU, and then displayed on the central display. The main task is to support the driver during parking by showing the surrounding of the car from a bird's eye perspective so that the driver can see the objects around the car. A common system model of this use case ensures, that the results of the simulation as well as the analytical approach are comparable. The architecture, the network topology, the used tasks, and the data dependencies are described in the following.

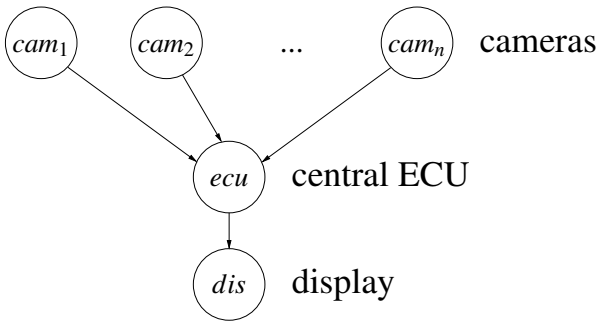


Figure 4.5.: The data dependency graph of the multi-camera system. The cameras send their captured images to the ECU which combines the different pictures into one output picture and sends this to the display.

Figure 4.5 shows the underlying data dependency graph. The cameras therefore record single frames and send them packetized via Ethernet to the central ECU. From the incoming packets of the different cameras, the single frames will be reconstructed. If all single frames belonging together are available at the ECU, the integrated bird's eye view will be calculated. Afterwards the picture will be packetized again and sent to the display. The display then reconstructs the picture from the incoming packet, stream, and displays the content.

The topology of the system is shown in Figure 4.6 and is based on a star network topology containing a central Ethernet switch. All components are directly interconnected by the switch. The switch itself consists of five 100 Mbit/s interfaces between the cameras and the display and one 1 Gbit/s interface to the central ECU. The switch backplane bus works with a maximum speed of 2 Gbit/s.

Camera Component The tasks of the camera are the recording and data compression of complete or partial frames and the creation of UDP packets containing the compressed images which are then sent to the central processing ECU. Figure 4.7(a) shows the setup of such a camera module cam_i which consists of the imager img_i, the processor cpu_i, as well as the Ethernet communication controller cc_i. After recording a single or partial frame, the imager interrupts the processor which then schedules the compression algorithm enc_i. After finishing the compression, the software network stack net_i is executed to packetize the information into one or more UDP packets p_{cam_i} and to send them to the central ECU. Since the execution times of the software task depends on the implementation of the software and the deployed ECU, a refer-

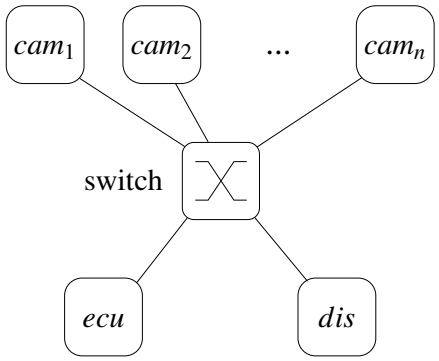


Figure 4.6.: The topology of the multi-camera system. The components are interconnected via a central Ethernet switch.

ence design was taken to measure the execution time. The send and receive network stack execution times $l_{net_i}(n)$ are dependent of the packet length n and are described by Equation (4.4). t_{fix} represents the fixed time interval which is independent of the packet length n. The variable fraction is denoted with $t_{var}(n)$.

$$l_{net_i}(n) = t_{fix} + t_{var}(n) \text{ ns} \tag{4.4}$$

Both software tasks are scheduled preemptive and priority-based. After the creation of each network packet p_{cam_i}, the Ethernet communication controller is triggered by a DMA access to send the packet.

Central ECU The processing of the different camera pictures is done within the central ECU. Figure 4.7(b) shows the hardware architecture and the mapped software tasks. It consists of the two resources 1.) the processor cpu_{ecu} and 2.) the Ethernet communication controller cc_{ecu}. The processor executes the send and receive network stacks *net* and the application dependent tasks decoder dec and integration *int*. Again, all tasks are scheduled in a preemptive manner and priority-based. After receiving a packet p_{cam_i}, the network stack unpacks the image data into memory and then the decoder dec is started. The decoder decompresses the image data. When all images of the different cameras are available, the integration task *int* is executed to calculate the bird's eye perspective. Finally, the resulting picture is packetized again and the packets p_{int} are send to the display.

Display Figure 4.7(c) shows the architecture of the display. It consists of an Ethernet communication controller cc_{dis}, a processor cpu_{dis} as well as the display dis. Again, the processor unpacks the incoming packets p_{int} and stores them into memory to provide the data to the display.

4.3.3. Evaluation Results

The following section focuses on the simulative results obtained by the discrete event simulator for the given use case of a multi-camera system. Additionally, the system end-to-end latency for fusion of one image frame is compared to the results obtained from an analytical approach. Since the simulator works on packet-level, detailed information about the system behavior can be determined as presented in the following.

Number of Packets Figure 4.8 shows the number of transmitted network packets per second for different packet lengths n. It is shown that the number of packets decreases when the packet length of the transmitted packets increases. In addition, it is possible to measure the drop rate of packets within each component by defining a particular buffer size b_s.

Resource Utilization The resource utilization of the different components for different packet lengths n is shown in Figure 4.9. Since each packet has to be processed by the software network stack, the resource consumption of the components increases when the packet length decreases and therefore more packets have to be processed by the software network stack. The results show that the resource utilization of the central ECU – the utilization of the task scheduler – grows up to 100 percent when reducing the packet length n of the transmitted packets down to 600 bytes. Packet lengths smaller than 600 bytes led to packet drops due to the overloaded processor within the central ECU.

End-to-End Latency Figure 4.10 highlights the system end-to-end latency denoting the duration from recording a sample by the camera to displaying the same sample on the display for different network packet lengths n. A comparison is made between the average latency determined by the simulative approach and the worst case latency derived from the analytical approach. The two times higher latencies of the analytical approach are caused by the facts that the simulation is typically not able

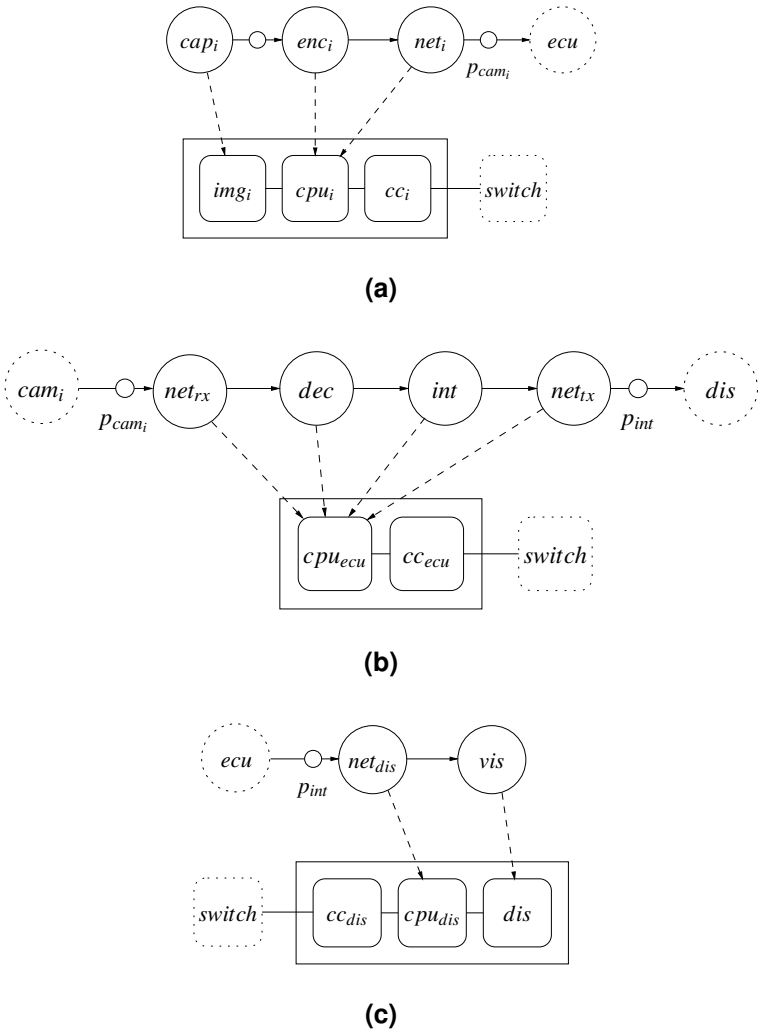


Figure 4.7.: (a) Shows the structure of the camera with the three components imager img_i, processor cpu_i, and communication controller cc_i and the three tasks 'image capturing' cap_i located at the imager, 'image encoding' enc_i, and the 'network stack' net_i. The tasks enc_i and net_i are both executed by the processor. The central ECU is presented in (b) and consists of the processor cpu_{ecu} and the communication controller cc_{ecu}. The software tasks 'network stack' net_{rx} and net_{tx}, 'image decoding' dec, and 'image integration' int are all executed by the processor. (c) Highlights the display component which consists of the communication controller cc_{dis}, the processor cpu_{dis}, and the display dis. The 'network stack' task net_{dis} is executed by the processor and the 'visualize image' task vis is located at the display.

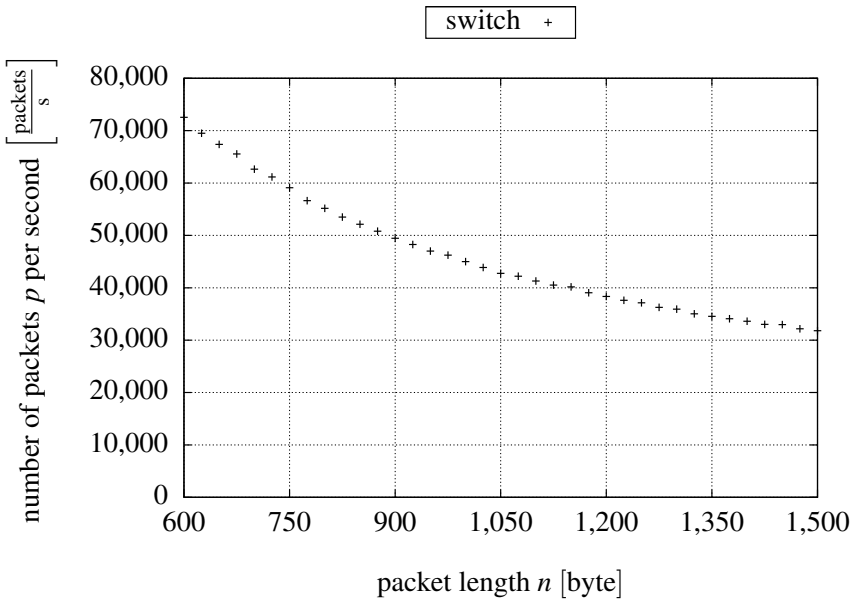


Figure 4.8.: The number of transmitted packets per second for different packet lengths n determined by the simulative approach.

to process all system states and the analytical model was not able to analyze all the details of the common model. A detailed description of the differences can be found in [RKH+10•].

4.4. Restbus Simulation and Evaluation of Switched Networks

This section addresses the question of how to test Ethernet-based automotive networks which are partly or completely available. Typically, not all devices of a distributed function are delivered at the same time during the development process, so it is necessary to test devices in simulated environments. The following physical Ethernet-Test-Switch concept was developed evaluate data rates, timings, as well as the functional behavior of Ethernet-networked devices. Finally, a restbus simulation concept is introduced to combine software simulation with existing hardware devices by using the Ethernet-Test-Switch. Throughput and latency measurements conclude the section.

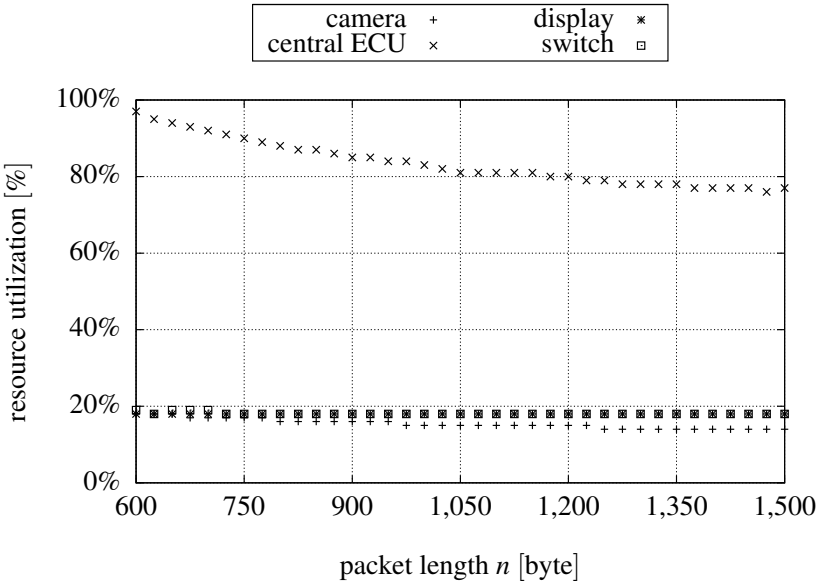


Figure 4.9.: The resource utilization of the different system components for different packet lengths n determined by the simulative approach.

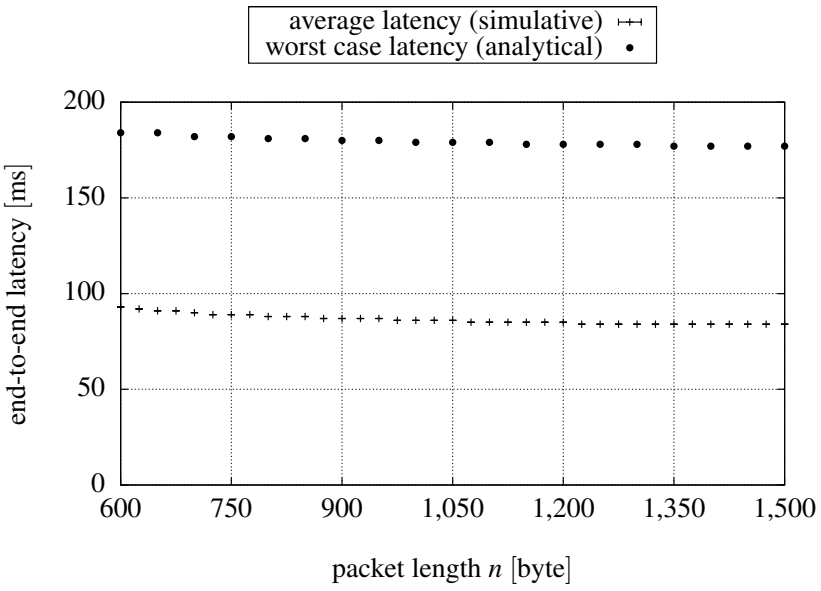


Figure 4.10.: The system-wide end-to-end latency denoting the duration from recording a sample by the camera to displaying the same sample on the display for different network packet lengths n. A comparison is made between the average latency determined by the simulative approach and the worst case latency derived from the analytical approach.

4.4.1. Ethernet-Test-Switch Concept

The following enumeration shows the basic requirements for an automotive Ethernet testing environment derived from current available automotive tools like CANoe [Vec11] from Vector or blue PiraT [Tel12] from Telemotive.

4.4.1.1. Requirements

The following list contains typical requirements or features of a network testing environment for automotive applications:

1. Hardware timestamping of packets
2. Determination of load utilization
3. Identification of network parameter settings
4. Monitoring of traffic
5. Generation of traffic
6. Possibilities for the use of network simulation

For example, hardware timestamping is used to measure the exact transmit or receive time of a packet or the time interval of packets belonging together. The determination of the load utilization can be used to compare the expected load utilization of a device with the actual load utilization. Network parameters can also be checked by test devices. In the case of Ethernet switches, for example, the address lookup tables can be compared with the estimated lookup entries. This is very helpful during the startup phase, when all connected devices start to send packets and the Ethernet switch learns the address entries of its different ports. The monitoring of traffic is typically used to check if all packets are send from a device and the packet generation can be used to stress devices. For example, the increase of the interrupt load of a receiving device.

4.4.1.2. Ethernet-Test-Switch

Since a standard Ethernet switch is not able to fulfill all the requirements above, the test device was moved into the network under test. The result is shown in Figure 4.11

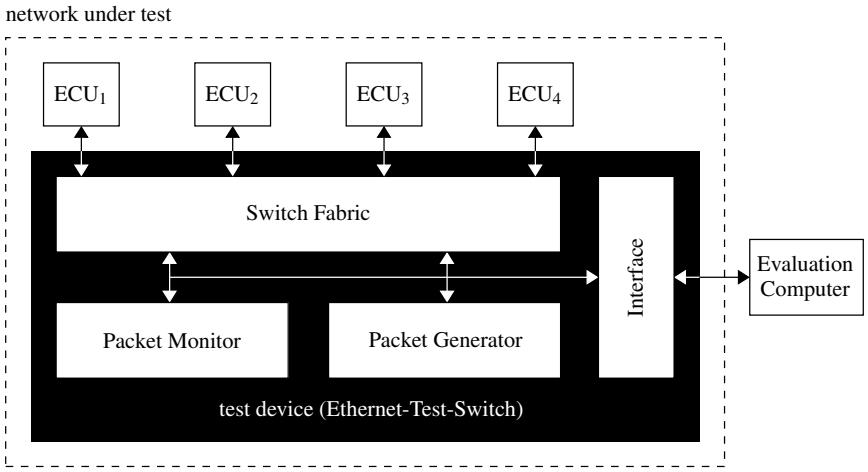


Figure 4.11.: An overview of the Ethernet-Test-Switch consisting of the components *Switch Fabric*, *Packet Monitor*, *Packet Generator*, and *Interface*.

where a standard Ethernet switch which is regularly part of the network under test is replaced by our Ethernet-Test-Switch. The Ethernet-Test-Switch concept is implemented on a *Field Programmable Gate Array* (FPGA) processor and consists of the components *Switch Fabric*, *Packet Monitor*, *Packet Generator*, and *Interface*. A more detailed implementation overview of the individual components is shown in Figure 4.12. It contains the implemented IP-blocks for one external Ethernet port. In general, the platform is designed to support a maximum of four external ports. Four additional ports can be added by using a second quad PHY extension board. The different components are described in detail in the following.

Switch Fabric The main tasks of the *Switch Fabric* are the switching of packets between different external ports and the maintenance of the lookup table. A buffered crossbar architecture [CL07] is used to switch packets between incoming and outgoing ports as shown in Figure 4.13. One advantage of this approach is that packets on disjoint paths through the switch can be switched at the same time without any interferences. In addition to the switching of packets, the actual utilization of the *Switch Fabric* like buffer fill levels and link utilizations is recorded and provided to the *Interface* component. Furthermore, it is possible to set or get the current lookup table or to disable the MAC learning algorithm at runtime. Finally, we added a hardware timer to get synchronized high-precision timestamps for incoming packets across all four

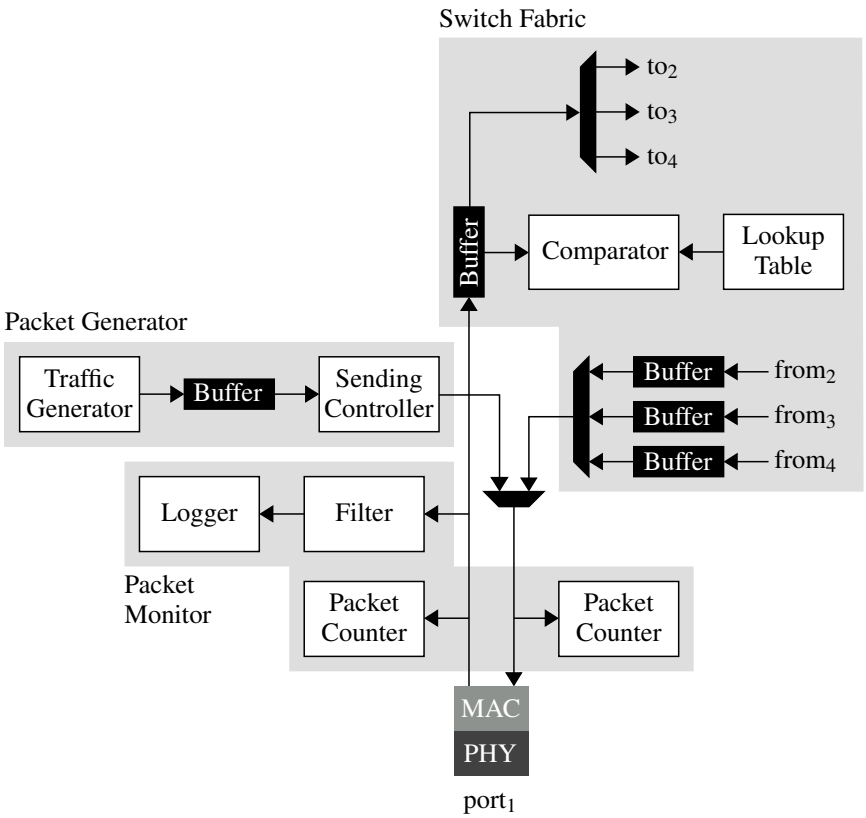


Figure 4.12.: A detailed implementation overview of the components *Switch Fabric*, *Packet Monitor*, and *Packet Generator* for one external port.

ports. The accuracy of the timestamps is 10 ns. With these capabilities, it is possible to meet the basic requirements (1) to (3).

Packet Monitor The *Packet Monitor* registers all incoming packets. Therefore, a log entry is created by the *Packet Monitor* for each incoming packet and sent to the *Interface*. Figure 4.14 shows the structure of such a log entry. It has a fixed length of 64 byte (minimal length of an Ethernet frame) independent of the length of the incoming Ethernet packet and contains several information about the packet like source and destination addresses, embedded protocols, or the routing path inside the switch. Moreover, it is possible to configure hardware filters to reduce the utilization of the *Interface* and to get prefiltered packet traces to increase the readability of the traces. The *Packet Monitor* fulfills requirement (4).

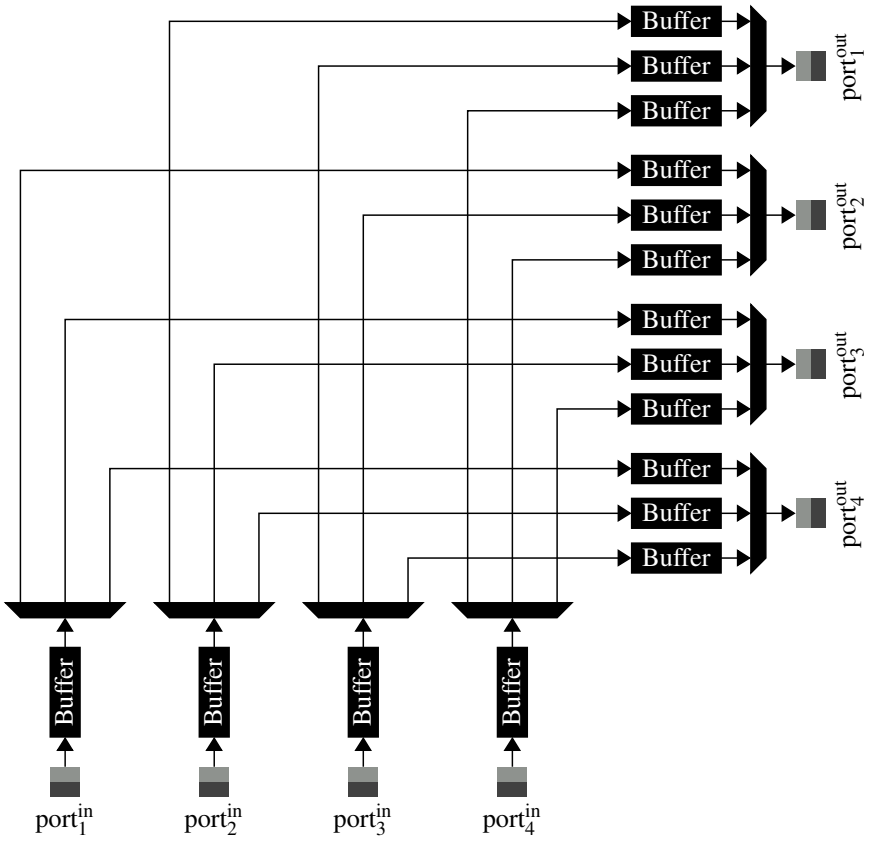


Figure 4.13.: The crossbar switching architecture of the *Switch Fabric*.

Packet Generator The main idea behind the *Packet Generator* is the generation of traffic at the ports of the Ethernet-Test-Switch in hardware. It is distinguished between cyclic and burst traffic. Cyclic traffic is generated periodically with a predefined cycle time and the number of packets to be sent can be limited or continuous. In burst mode, a defined number of packets are sent one after the other without interruption. These mechanisms can be used to test systems by adding additional traffic to the system under test. Each port has its on generator and the packets created by it are preferred by the outgoing ports. This means, that a packet created by the packet generator is always send immediately to the Ethernet MAC. Requirement (5) is therefore implemented by the *Packet Generator*.

Interface The *Interface* component is used to bundle the whole communication between the other components and the evaluation computer in a structured way. Therefore, different packet types are specified for the different components. An example is

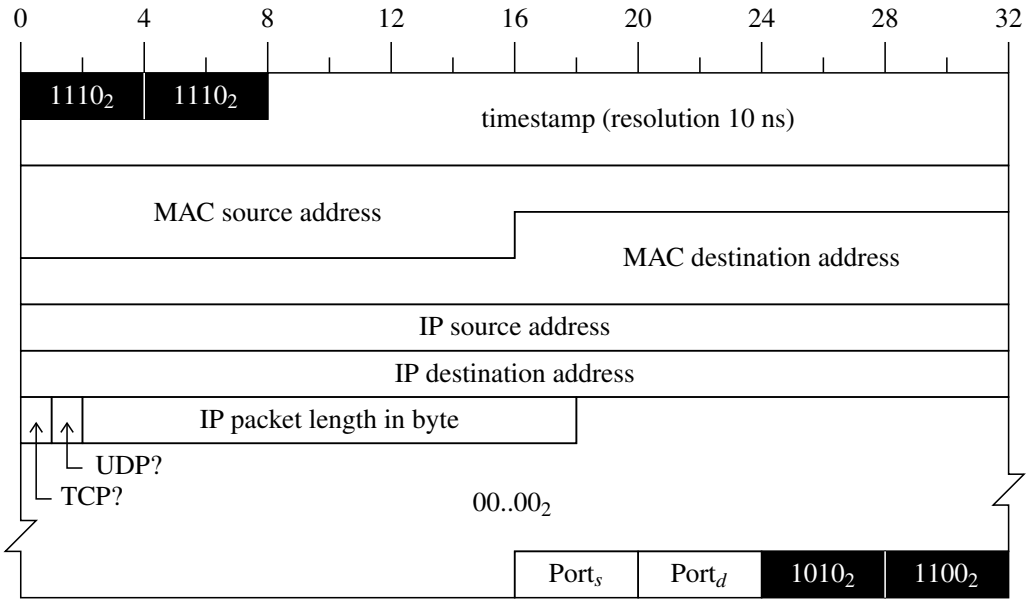


Figure 4.14.: The 64 byte long data structure of the log entry created for each incoming packet.

the 64 byte long log packet shown in Figure 4.14.

4.4.2. Restbus Simulation

The basic consideration is to interconnect existing Ethernet-based devices with simulated Ethernet-based devices. Figure 4.15 presents the structure of the simulation environment. The left side of the figure contains the extended Ethernet-Test-Switch with four existing devices connected to it by the hardware ports MAC^1 to MAC^4. The right side shows an evaluation computer running a simulation software which is simulating four more devices with the virtual ports MAC^5 to MAC^8. A Software *Application Programming Interface* (API) running on the evaluation computer connects the simulator with the Ethernet-Test-Switch *Interface* and is the basis for exchanging packets between each other. In addition, it was necessary to extend the *Switch Fabric* to support additional virtual ports MAC^5 – MAC^8 which are assigned to predefined MAC addresses. Whenever an incoming packet arrives with a destination address of a virtual port, the packet will be forwarded to the *Interface* component which tunnels the packet via the interface connection MAC^i to the evaluation computer. The

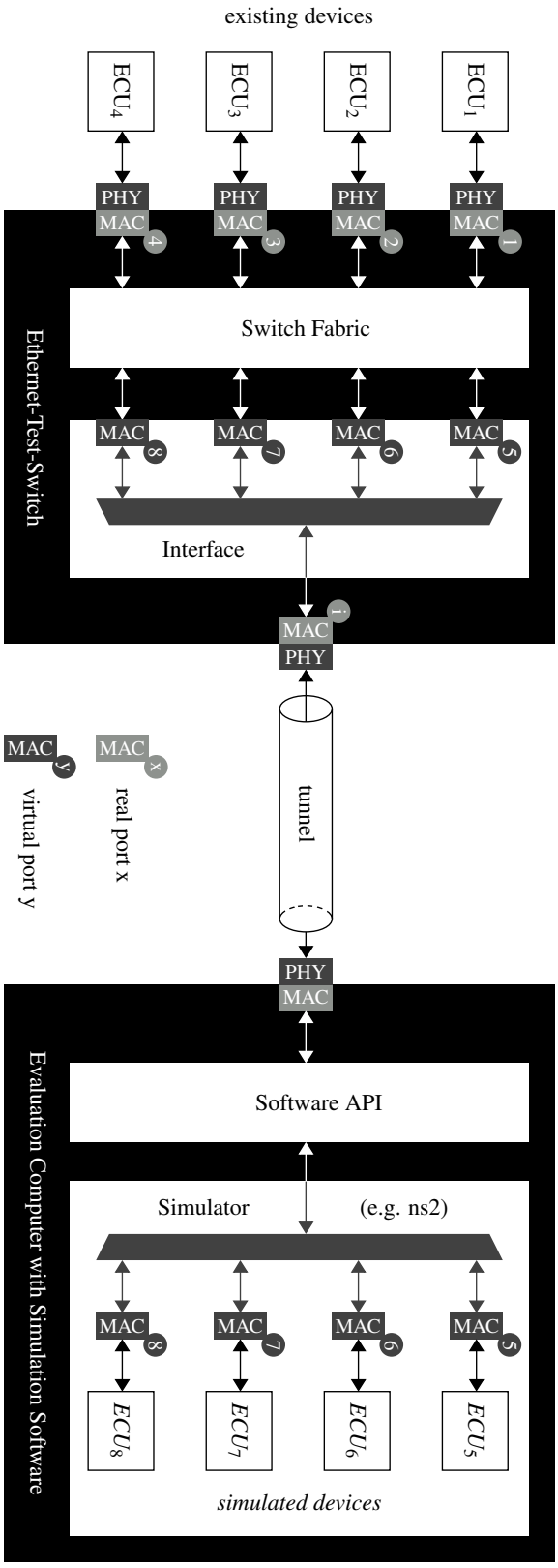


Figure 4.15.: The proposed concept for network simulation of mixed physical and virtual devices. In this case, four existing ECUs ECU_1 to ECU_4 are interconnected via the Ethernet-Test-Switch to four simulated ECUs ECU_5 to ECU_8. The *Interface* components connect the Ethernet-Test-Switch to the computer running the simulation software.

Software API receives the packet and forwards it to the simulation software. Thus, it is feasible to receive Ethernet frames from existing devices within a simulation software. Similarly, it is possible to generate Ethernet packets through the simulation software to forward them to existing devices. Such a packet is sent via the Software API to the Ethernet-Test-Switch *Interface*. The *Interface* then forwards the packet with the lookup table to the desired external port and the device will receive the packet.

In summary, it is now possible to interconnect existing physical Ethernet-based devices with simulated Ethernet-based devices to build up an integrated environment for testing mixed Ethernet-based systems. Requirement (6) is therefore fulfilled.

4.4.3. Experimental Results

This section presents implementation details, some of the evaluation measurements, and analytically determined limitations of our implementation of the Ethernet-Test-Switch and the Software API. The performance measurements were carried out with Spirent Testcenter [Spi11] for packet generation and monitoring and a Lenovo Thinkpad T500 with Windows XP as an evaluation computer running the Software API.

Implementation Details The Ethernet-Test-Switch is implemented on an Altera Stratix III FPGA development board and uses a Marvell Quad-PHY 88E1145 (PHY_1 – PHY_4) connected via a HSMC interface. The connection to the PC is realized with an on-board Marvell Single-PHY 88E11111 (PHY_i). Except for the Ethernet MAC controller from Altera, all other components are self-created VHDL *Intellectual Property blocks* (IP-blocks). Table 4.1 summarizes the resource utilization of the Altera Stratix III EP3SL150F1152C2N device. The Software API is written in C and released as a *Dynamic Link Library* (DLL) for Windows.

Switch Fabric For measuring the maximum packet throughput and the minimum, average, as well as maximum latency of the Switch Fabric, two ports of the Testcenter were connected to the Ethernet-Test-Switch. Packets of different packet sizes and loads were sent from one port to the other. Detected packet losses at the receiving port of Testcenter indicates an overload of the Switch Fabric. The measured latency is the average latency of all transmitted packets. Figure 4.16 summarizes

the measured throughput and latency results of the Switch Fabric. The throughput b_{raw} of the Switch Fabric is 999.994 Mbit/s which is near line rate and independent of the packet size p_l. Note that currently 100 Mbit/s are discussed for Automotive-Ethernet. For the latency, delays between 1.8 µs and 17.1 µs for different packet sizes were measured. The latency $\tau_p(p_l)$ of a packet p of length p_l byte is divided into the duration $\tau_{store}(p_l)$ to store an incoming packet within the Switch Fabric and the duration $\tau_{forward}$ to forward the packet to an outgoing port. Equation 4.5 calculates the latency of the Switch Fabric, where n denotes the length of the packet in byte. In addition, we compared our latency measurements with standard Ethernet switches and obtained similar latency results.

$$\tau_p(p_l) = \tau_{store}(p_l) + \tau_{forward} \tag{4.5}$$

Packet Monitor In general, the performance of the packet monitor is limited by the available throughput between the Ethernet-Test-Switch and the evaluation computer. The higher the throughput is, the more monitoring packets p_m can be transmitted during a time interval. Equation 4.6 calculates the number of monitoring packets $N_{p_m}(b_i)$ per second that can be transmitted over the interface to the evaluation computer, where b_i denotes the available throughput at interface connection (PHY_i) in Mbit/s and l_{pre}, l_{p_m}, and l_{ifg} are the lengths of the preamble, monitoring packet, and inter frame gap in byte.

$$N_{p_m}(b_i) = \frac{b_i \cdot 10^6}{\left(l_{pre} + l_{p_m} + l_{ifg}\right) \cdot 8} \tag{4.6}$$

Table 4.1.: The resource utilization of the Altera Stratix III FPGA containing the Ethernet-Test-Switch.

logic utilization	32%
dedicated logic registers	25,156 / 113,600 (22%)
total pins	126 / 744 (17%)
total block memory bits	2,809,320 / 5,630,976 (50%)

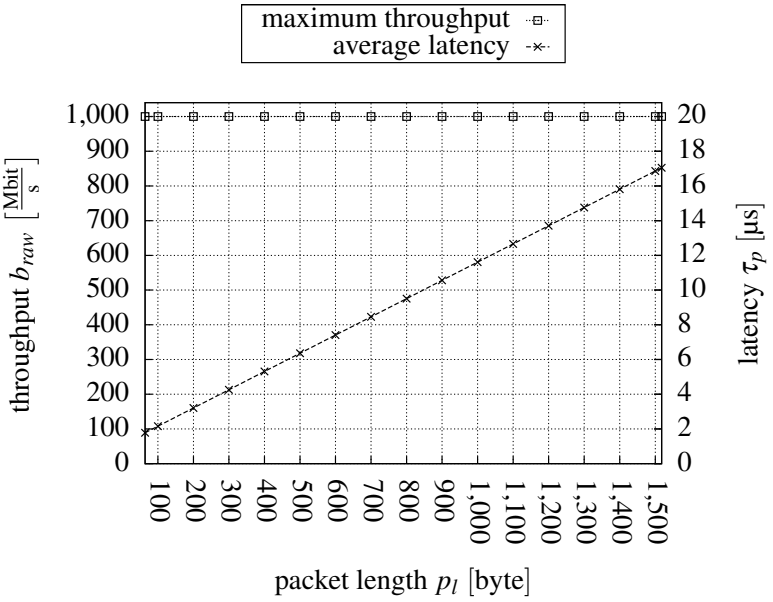


Figure 4.16.: The maximum measured throughput b_{raw} and the average measured latencies τ_p of the Switch Fabric for different packet length p_l.

With $b_i = b_{raw} = 999.994$ Mbit/s and $l_{p_m} = 68$ byte (including the 4 byte cyclic redundancy check), the number of monitoring packets per second that can be transmitted is about $1,420,446$. The maximum incoming throughput $b_m(n, b_i)$ caused by packets p arriving at the four switch ports with packet size n that can be handled by the packet monitor is calculated by Equation 4.7. Figure 4.17 shows the maximum incoming throughputs for four ports $b_m(n, b_{raw})$ depending on different packet sizes n in byte.

$$b_m(n, b_i) = N_{p_m}(b_i) \cdot \left(l_{pre} + n + l_{ifg}\right) \cdot 8 \tag{4.7}$$

The measurement results presented in Figure 4.18 show much smaller throughputs caused by the limited performance of the network interface of the evaluation computer and the software to display the monitored packets. With a minimum packet size n of 64 byte, a throughput b_m of 20 Mbit/s could be achieved. The maximum packet size n of 1518 byte led to a throughput b_m of about 400 Mbit/s.

Packet Generator The output throughput b_g of the packet generator depends on the applied operation mode. When working with preconfigured packets in the cyclic or burst mode, the full throughput b_g^{pre} of b_{raw} = 999.994 Mbit/s can be utilized. If an individual packet p_i shall be send, it will be necessary to configure the packet generator for each outgoing packet. The resulting throughput of outgoing packets b_g^{in} in Mbit/s is shown in Equation 4.8 for different packet sizes n and limited by the configuration time t_{cfg}, $t_{p_i}(n)$ the duration to transmit the packet from the evaluation computer to the packet generator, and $t_{send}(n)$ the duration to send the packet by the packet generator. The measured throughputs b_g^{real} of a real application are much lower conditioned by higher software execution times. Figure 4.18 summarizes the results obtained from the packet generator. In real situations, a minimum throughput of 4 Mbit/s was measured with a minimum packet size of 64 byte. The maximum throughput of 59 Mbit/s was reached with a packet size n of 1518 byte.

$$b_g^{in}(n) = \frac{n \cdot 8}{t_{cfg} + t_{p_i}(n) + t_{send}(n)} \tag{4.8}$$

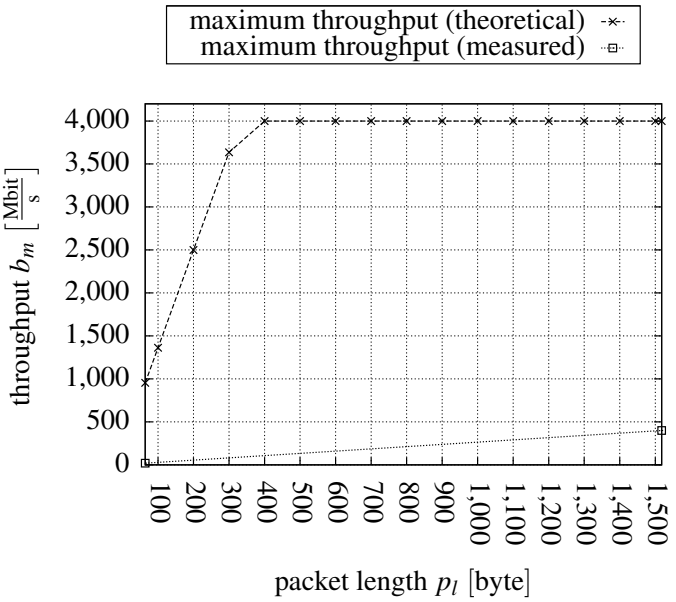


Figure 4.17.: The theoretical maximum and measured maximum incoming throughputs at four ports b_m that can be handled by the packet monitor for different packet sizes n.

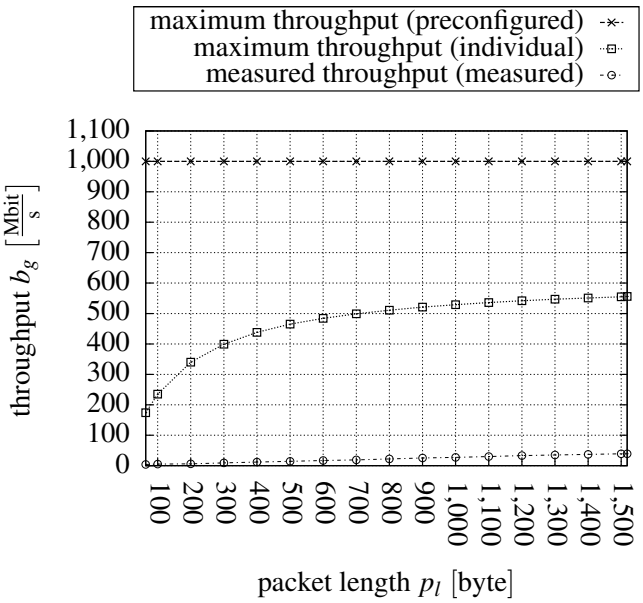


Figure 4.18.: The maximum measured throughputs b_g^{pre} for preconfigured packets, the theoretical maximum throughputs b_g^{in} for individual packets, and the maximum measured outgoing throughputs of individual packets send by an application for different packet sizes n.

4.5. Summary

The 1-10-100 rule shows the necessity to have a comprehensive test and evaluation tool chain to find errors in the early stages of the design phase and to minimize the residual error probability at the start of production. This chapter introduced two fundamental methods to simulate and test Ethernet/IP embedded systems.

For the first approach a case study based on a multi-camera embedded system which was interconnected via Ethernet/IP was presented. By adapting a discrete event simulator with Ethernet simulation modules and automotive-specific parameters, the use case could be evaluated without having any corresponding hardware available. The results showed, that the chosen network architecture is able to transport the packets between the different components as long as the packet length is greater than 600 byte per packet. In this cases no packet loss has occurred and the timing requirements were met. Packet lengths smaller than 600 byte led to a resource conflict within the central ECU due to too many packets which have to be processed by the software network stack and thus leading to loss of image information. In addition, the end-to-end latency could be calculated to determine the portion of the network latency compared to the overall end-to-end latency and compared to the results of an analytical approach which calculates the worst case latency.

The second concept dealt with the problem of traffic monitoring, traffic generation, and restbus simulation for switched Ethernet/IP networks. Here, a concept is proposed and designed which extends a standard Ethernet switch by adding a transparent switch fabric and additional modules to disclose the internal switch mechanism to an external evaluation tool. With this approach, it was possible to trace the complete switch traffic without using port mirroring. In addition, highly precise timestamps are added to reconstruct the timing behavior or ordering of packets. A packet generator enabled stress tests for connected components by creating packets with different payloads and throughputs. Finally, an application programming interface facilitates the switch to provide restbus simulation by connecting a simulation framework, as presented above, for example, to the Ethernet-Test-Switch. The evaluation of the switch itself shows a measured packet monitor throughput of about 400 Mbit/s limited only by the connected evaluation computer. The packet generation performs up to line rate. In addition, it could be shown that restbus simulation can be performed with a throughput of up to 4 Mbit/s. For simple interaction testing or the testing of

functional behavior, this throughput is sufficient. For more complex simulations including video or audio samples, for example, a more performant software simulation, which was the bottleneck in this case, is needed.

5. Conclusions and Future Work

In the introduction, it was motivated that the number and complexity of distributed electric and electronic systems within vehicles will further increase in the future. This leads to more complex networking requirements for the underlying E/E-architecture. Examples are high data rate video or data sources or networked vehicles with internet access and the ability to connect external devices like notebooks or media players. One way to meet the upcoming infrastructure requirements is the introduction of Ethernet as a high-speed vehicle network together with the IP protocol family which supports a perfect separation of the different hardware and software layers by the usage of defined interfaces. Ethernet is already in discussion for different use cases like vehicle diagnostics, software download to ECUs, or as a charging interface for electrical vehicles. It is also in discussion for internal communication like high speed camera links or when designing future vehicle network architectures – the Ethernet backbone architecture, for example. IP offers logical addresses and higher layer transport protocols like UDP for datagram communication or TCP as a reliable connection between two end nodes. The usage of globally unique and internet-compatible network addresses offers general access to the vehicle from the internet and a multitude of new applications like intelligent routing or payment systems, for example. To introduce such a new communication technology together with its higher layer logical protocols, the automotive requirements have to be met and typical use cases like gateway interfaces to other technologies have to be evaluated. Today, Ethernet is established in many different sectors like the internet, consumer electronics, telecommunication, industrial automation, as well as in the avionic sector but not within the automotive sector for vehicle internal communication. In order to fulfill the automotive requirements and use cases, the work of this thesis concentrates on fundamental research topics like the robustness and efficiency of Ethernet and IP, the gateway capabilities to communicate with already established communication technologies, and the testing of Ethernet-networked automotive applications during the design and devel-

opment phase.

This thesis presented concepts in the area of 1) *Technology Safeguarding of Ethernet/IP-based Automotive Communication Networks*, 2) *Integration and Migration Concepts for Ethernet/IP-based E/E-Architectures*, and 3) *Simulation and Pre-production Testing of Ethernet/IP-based E/E-Architectures*.

Contributions to Technology Safeguarding of Ethernet/IP-based Automotive Communication Networks In order to introduce Ethernet as an additional automotive communication technology when designing future network E/E-architectures, Ethernet has to fulfill general communication technology requirements. Chapter 2 presented concepts 1) to determine the error rates of given Ethernet implementations, 2) to measure the synchronization accuracy of time synchronized end nodes under different climate conditions, 3) and to optimize the data throughput when using low cost resource-restricted processors. In the area of error rate analysis, an error rate determination concept for Ethernet PHYs was proposed. The goal is to determine the bit and packet error rates as well as the residual error rates for different protocol error detection mechanisms. In addition, the overall bit error distribution and the number of bit errors per packet could be determined. BCI and vehicle measurements showed novel bit and packet error results for given Ethernet PHY implementations. In order to fulfill extended automotive temperature requirements, a concept was designed to measure the time synchronization accuracy of distributed nodes under different climate conditions. Industrial climate chambers were used to stress the synchronization end nodes. For the synchronization, the packet-based algorithm specified in IEEE 802.1AS was implemented together with Ethernet as underlying communication technology. For several days, exhaustive measurements produced a large set of results and showed that the packet-based IEEE 802.1AS algorithm is able to work under different temperature conditions. A temperature range from $-10\,°\mathrm{C}$ ($+14\,\mathrm{F}$) to $+70\,°\mathrm{C}$ ($+158\,\mathrm{F}$) was evaluated. Finally, extreme shock temperature measurements rapidly changed the environment temperature by $80\,\mathrm{K}$ in less than 5 seconds. In summary, it could be shown that IEEE 802.1AS and Ethernet are able to provide highly-accurate distributed synchronized timers in the nanosecond range. Finally, a concept was developed to optimize the transmit throughput of an Ethernet-based communication link. Different variants used software adaptions as well as hardware support to increase the throughput up to Gigabit Ethernet line-rate

based on an 85 MHz FPGA-based processor were proposed. In summary, it could be shown that the throughput measured in a non-optimized environment can be enormously increased.

Contributions to Integration and Migration Concepts for Ethernet/IP-based E/E-Architectures Based on the positive results of the fundamentals above, gateway and replacement approaches have been designed to enable the connection between existing technologies like CAN or FlexRay networks with Ethernet/IP. In addition, concepts to completely replace a CAN or FlexRay network by Ethernet and IP have been proposed. In the gateway area, a CAN/Ethernet gateway concept was designed, prototypically implemented, and finally tested with self-created input patterns and within an E-class vehicle. Beside the technical feasibility, detailed evaluations have been made: In particular, it was shown that the average end-to-end latency over two gateway ranged from 500 µs in case of a one-to-one mapping without buffering up to 4500 µs in case of a buffered approach. The average throughput without buffering was about six times higher than the CAN utilization. Due to the use of buffering, the Ethernet data rate could be reduced down to twice the CAN utilization. The maximum resource consumption of the gateway processor was about 27 %. In addition, a FlexRay/AVB gateway concept was introduced to forward data of the static segment of FlexRay to Ethernet AVB and vice versa. The transformation concept covered in-cycle and multiple-cycle buffer approaches to replace typical transport protocols for FlexRay. Based on a real FlexRay configuration, it could be shown that the resulting Ethernet AVB throuphput was about three times higher compared to the FlexRay resource utilization. Moreover, long-term replacement approaches for CAN and FlexRay were designed and evaluated. In the case of CAN, the packetizatíon optimization methods were proposed based on a reference one-to-one mapping, an EA-based approach, and finally a rule-based approach. The results reveal worst case data rates by assuming asynchronous messages as cyclic messages with defined debounce times. Moreover, it was shown that the resulting worst case data rate may be about 100 times higher compared to CAN resource utilization. The EA-based und rule-based approches was able to half the data rates in the Ethernet network. In the case of FlexRay, the FlexRay gateway concept was adopted by mapping the different conncetions to distinct VLANs. A test setup showed that a send data rate within the Ethernet AVB network was about three times higher than the FlexRay resource

utilization. With buffering, the send data rate could be reduced to be close to the FlexRay data rate. Furthermore, the average latency was comparable to the latency caused by the FlexRay network. In summary, this chapter presented four extensive concepts for interconnecting or replacing established technologies with Ethernet/IP. Prototypical implementations confirmed the technical feasibility even in a current E-class vehicle.

Contributions to Simulation and Pre-production Testing of Ethernet/IP-based E/E-Architectures Testing plays an important role to ensure the availability and correctness of the different integrated applications. In order to evaluate Ethernet-networked systems in an early stage of development when no hardware or software is available yet, a concept was designed to simulate Ethernet-based automotive networks. Furthermore, a concept was proposed to test Ethernet-based networks which are partly or completely available. In the area of simulation, a discrete event simulator was extended with Ethernet modules to enable the simulation of the timing as well as the functional behavior of automotive embedded systems. Ethernet transceivers, controllers, and switches were modeled within the simulator. In addition, a task model was introduced to simulate the behavior of the software network stack. Based on this extensions, a multi-camera embedded system case study was evaluated. For this use case, it could be shown, that the chosen network architecture is able to transport the packets between the different components as long as the packet length is greater than 600 byte per packet. Packet lengths smaller than 600 byte led to a resource conflict within the central ECU due to too many packets which have to be processed by the software network stack. The simulation showed that the amount of resources within the central ECU are insufficient. In this case, the task scheduler was not able to schedule all necessary tasks. In addition, the end-to-end latency could be calculated to determine the portion of the network latency compared to the overall end-to-end latency. The second concept dealed with the capturing, insertion, and restbus simulation of switched Ethernet/IP networks. For this purpose, a concept was designed which represented a standard Ethernet switch by using a transparent switch fabric and additional modules to disclose the internal switch mechanism to an external evaluation tool. With this approach, it was possible to trace the complete switch traffic without using port mirroring. In addition, highly precise timestamps were added to reconstruct the timing behavior or ordering of packets. A packet generator

enabled stress tests for connected components by creating packets with different payloads and throughputs. Finally, an application programming interface facilitates the switch to provide restbus simulation by connecting a simulation framework, as presented above for example, to the Ethernet-Test-Switch. The evaluation of the switch itself showed a measured packet monitor throughput of about $4,000$ Mbit/s limited only by the connected evaluation computer. The packet generation performed up to line rate and the restbus simulation allowed throughputs of up to 4 Mbit/s between physical and virtual devices carrying simulated information.

5.1. Future Work

Possible future research areas are: 1) *Ethernet/IP-based Communication Matrix Design and Storage*, 2) *Automotive Usage of Higher-Level Protocols*, and 3) *Timing Analysis*. In the communication area, Ethernet, IP, and application-related file formats have to be designed to describe the Ethernet topology and the communication between the different ECUs. Beside the description of the communication interfaces at ECU level, new operation system specifications like AUTOSAR [AUT12] do also consider the interface descriptions on *Software Component* (SWC) level. In this context, the mapping between ECU and SWC interfaces has to be considered. In addition, gateway configuration tables have to be extended to support Ethernet/IP. The goal should be a set of file formats and specifications which are OEM and supplier-independent. Another important topic is the understanding of the different higher layer protocols together with resource-restricted processors and memories. Todays network stack implementations, for example, use their own buffers and administration tables. Another point is the interaction between the different higher layer protocols. It is necessary to ensure that the different protocols can work in parallel without any conflicts. When implementing time-critical or real-time applications, timing analysis becomes an important field. When designing control systems, for example, worst case latencies are typically assumed. In such a case, it is necessary to ensure that the latencies of latency-sensitive packets do not exceed the specified maximum values. A detailed overview of timing analysis for current communication technologies in the automotive context can be found in [ST12].

A. German Part

Ethernet und IP für E/E-Architekturen im Automobil – Technologieanalyse, Migrationskonzepte und Infrastruktur

Zusammenfassung

Aktuelle Premiumfahrzeuge enthalten eine Vielzahl von verteilten Kundenfunktionen, welche alle Bereiche des Fahrzeugs wie beispielsweise *Antrieb*, *Fahrwerk*, *Innenraum* und *Fahrerassistenz* abdecken. Ein Großteil dieser Systeme wird hierbei mittels einer gemeinsam genutzten Netzwerkinfrastruktur, ein Teil der sogenannten *Elektrischen und Elektronischen Architektur* (E/E-Architektur) des Fahrzeugs, vernetzt. Die E/E-Architektur umfasst neben der reinen Kommunikation innerhalb des Fahrzeugs auch die Verbauung von Komponenten und die Abbildung von Kundenfunktionen auf deren logischen und physikalischen Komponenten, um nur einige weitere Bestandteile zu nennen. Für die Kommunikation stehen unterschiedlichste Kommunikationstechnologien und sogenannte *Gateways* zur Verfügung, welche die Kommunikation zwischen mehreren Technologien ermöglichen. Typische Kommunikationstechnologien für die Vernetzung von verteilten *Steuergeräten* (SG) sind hierbei *Local Interconnect Network* (LIN) [LIN10], *Controller Area Network* (CAN) [CAN91], FlexRay [Fle10], *Media Oriented Systems Transport* (MOST) [MOS10] und *Low Voltage Differential Signaling* (LVDS) [LVD95]. Die allgemeinen Vorzüge einer gemeinsamen Kommunikationsinfrastruktur sind unter anderem die Wiederverwendbarkeit von Informationen einzelner Komponenten für unterschiedlichste Funktionen, die Reduzierung des Verkabelungsaufwandes (beispielsweise die Vermeidung paralleler Kommunikationsleitungen bei gleichen Verbauräumen), die einfache Erweiterbarkeit um neue Komponenten und die Unterstützung von unterschiedlichen Ausbaustufen für die effiziente Abbildung von Fahrzeugvarianten.

Forschungs- und Vorentwicklungsaktivitäten einiger *Automobilhersteller* (OEMs) deuten darauf hin, dass sowohl die Anzahl als auch die Komplexität solcher verteilter Funktionen in Zukunft weiter zunehmen wird. Dies führt zu komplexeren Anforderungen an die darunter liegende E/E-Architektur und deren Kommunikationstechnologien [SKB+11•]. Abbildung A.1 zeigt beispielsweise die Entwicklung der Kommunikationsanteile von E/E-Architekturen der Mercedes Benz E-Klasse von

Mercedes Benz. Es zeigt auf, dass die Anzahl der Steuergeräte, Busse, und Signale in den vergangenen Jahren deutlich angestiegen sind und das die Anforderungen an die Datenraten welche innerhalb des Automobils zu übertragen sind aufgrund komplexer Sensoren drastisch ansteigen werden. Beispiele sind hier Objekt-basierte Radarsensoren oder Videokameras.

Ethernet, das *Internet Protocol* (IP) [Req81a] und darüber liegende Transportprotokolle wie beispielsweise das *User Datagram Protocol* (UDP) [Pos80] oder das *Transport Control Protocol* (TCP) [Req81c] könnten eine Lösung darstellen, um die zukünftigen Anforderungen an die E/E-Architekturen der nächsten Generation zu erfüllen. Ethernet ist durch die *Institute of Electrical and Electronics Engineers* (IEEE) 802.3 Arbeitsgruppe [IEE12b] standardisiert und besteht aus mehreren Standards welche unterschiedliche physikalische Medien und Übertragungsgeschwindigkeiten unterstützen. Ebenso sind die meisten darüber liegenden Protokolle wie IP und UDP quasi standardisiert durch unterschiedliche *Request for Comments* (RFC).

Die Hauptgründe für die Betrachtung von Ethernet/IP[1] im Fahrzeug sind, dass Ethernet/IP bereits für einige Anwendungen im Automobil gesetzt ist. Beispielsweise für Kommunikation von Fahrzeug und Außenwelt. Die Verwendung von Ethernet/IP für fahrzeuginterne Kommunikation wird zudem innerhalb der Automobilindustrie diskutiert. Letztlich stellt die mögliche Übernahme von Technologieanteilen aus der Verbraucher- und Industrievernetzung in Richtung eines Automobilstandards für Ethernet eine weitere attraktive Möglichkeit dar.

Beispiele für die Kommunikation von Fahrzeug und der Außenwelt sind die Fahrzeugdiagnose, die Fahrzeug-zu-Fahrzeug-Kommunikation, das Einspielen von Software-Updates und das Laden von elektrischen Fahrzeugen an der Ladestation. Im Diagnoseumfeld wird derzeit ein neuer ISO-Standard entwickelt, nämlich *Diagnostics over Internet Protocol* (DoIP) [ISO11a]. Dieser spezifiziert Ethernet und IP als einheitliche Kommunikationsschnittstelle zwischen dem Fahrzeug und dem Tester. Im Bereich der Vernetzung von Fahrzeugen untereinander oder der Vernetzung von Fahrzeugen und Umgebungsobjekten ist IP als einheitliches Kommunikationsprotokoll gesetzt [Car12]. Externe Partner können beispielsweise andere Fahrzeuge, Ampeln, oder weiter Infrastrukturobjekte sein. Die Nutzung von Ethernet für das Einspie-

[1] Im Folgenden dient der Begriff Ethernet/IP als Sammelbegriff für die Technologien und Protokolle IEEE 802.3, IEEE 802.1, IP und weitere darüber liegende Protokolle wie beispielsweise UDP, TCP oder IEEE 1722.

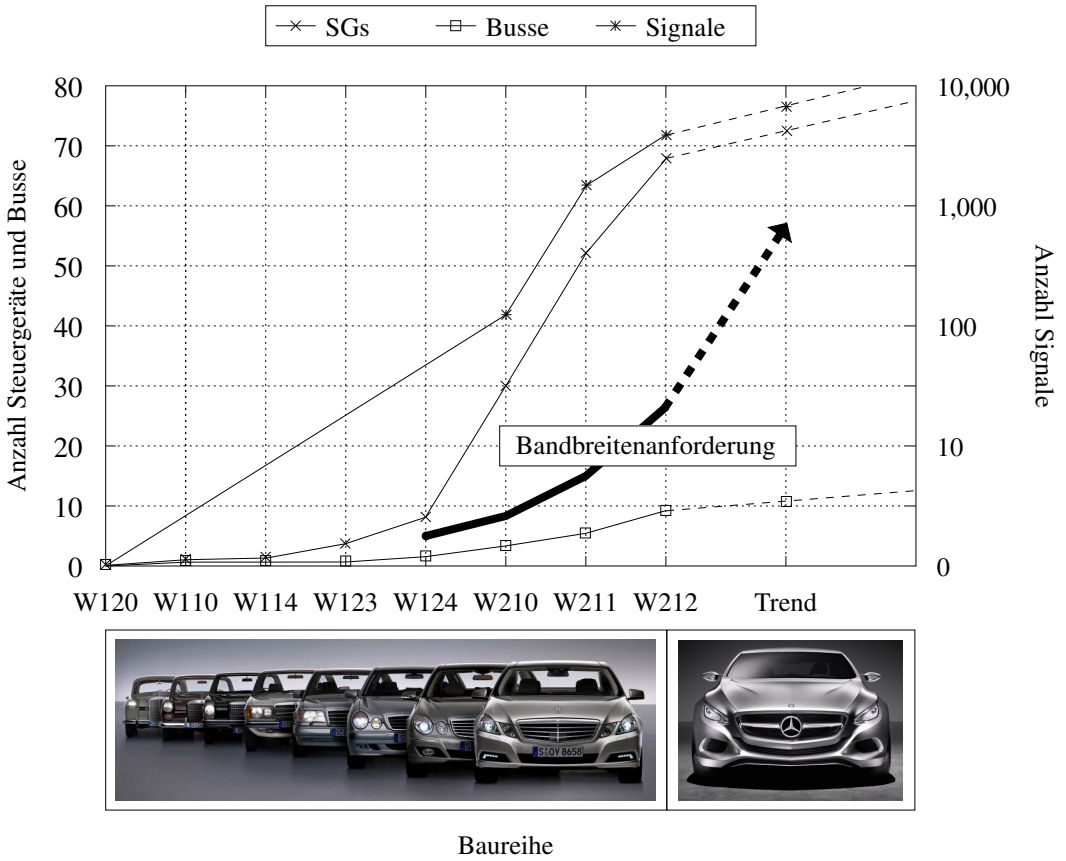


Abbildung A.1.: Die Entwicklung der Kommunikationsanteile der E/E-Architekturen der Mercedes Benz E-Klasse. Dargestellt sind die Anzahl der Steuergeräte, Busse und Signale für die jeweilige Baureihe. Zusätzlich wird der zukünftige Trend skizziert und die erwarteten Anforderungen an die Datenraten werden aufgezeigt [Rei10].

len von Software-Updates beschleunigt die Updatezeiten drastisch, da der zentrale Flaschenhals, nämlich die Verbindung zwischen Tester und Fahrzeug entfällt. Heutige Lösungen oder Lösungsansätze sind hier jedoch automobilherstellerspezifisch. Powerline [IEE10a] und IP sind außerdem in der Diskussion für die Ladeschnittstelle zwischen Fahrzeug und Ladestation. Benötigt werden hier Bezahlsysteme, Internet-Zugang und weitere Anwendungen während des Ladens. Hier hat sich der Sammelbegriff *Smart Charging* [ISO11d] etabliert.

Im Bereich der fahrzeuginternen Kommunikation wird Ethernet für die Anbindung von Kameras standardisiert und die Eignung als Ethernet-Backbone für alterna-

tive Fahrzeugvernetzungsarchitekturen wird grundlegend untersucht. Mögliche Vorteile für den Einsatz von Ethernet zur Vernetzung von Kameras sind Kostenvorteile durch die Nutzung von ungeschirmten Leitungen und die Verfügbarkeit von Switches zur Bildung von komplexeren Topologien im Gegensatz zur direkten Vernetzung zwischen jeweils zwei Steuergeräten wie im Fall von LVDS. Neben den Vernetzungstechnologien aus der Infotainment-Domäne mit dessen hohen Datenraten, könnte die limitierte Datenrate der heutigen automobilen Kommunikationstechnologien von 10 MBit/s ein weiterer Grund für die Einführung von Ethernet zur Kommunikation innerhalb einer Fahrzeugdomäne sein. Zudem wird Ethernet für die Einführung von alternativen Fahrzeugarchitekturen untersucht. Beispielweise die sogenannte Backbone-Architektur [BRR11], in der Ethernet als Hochgeschwindigkeitsnetzwerk genutzt wird, um die unterschiedlichen Fahrzeugdomänen miteinander zu verbinden.

Ein weiterer wichtiger Punkt ist die Untersuchung der Adaptierbarkeit von existierenden Ethernet-Technologien aus den unterschiedlichen Industriebereichen. Die Adaption existierender Technologien ermöglicht Kostenersparnisse und verkürzte Entwicklungszeiten vor allem im Vergleich zu einer kompletten Neuentwicklung eines automobilspezifischen Fahrzeugnetzwerks. Derzeit findet Ethernet in der Unterhaltungselektronik, Industrieautomatisierung, Telekommunikation und in der Luftfahrt seinen Einsatz. Es wurden hierzu mehr als ein Duzend Lösungen wie beispielsweise [Tim12, Eth12a, Pro12, Eth12c, Ser12] entwickelt um den spezifischen Anforderungen wie harte Echtzeit oder erhöhte Anforderungen gegenüber Umwelteinflüssen gerecht zu werden.

Aus der technischen Sicht ist Ethernet und IP eine ebenso attraktive Alternative. Ethernet bietet unterschiedliche Datenraten von 10 MBit/s bis 40 GBit/s für Anwendungen im nicht automobilen Umfeld. Darüber hinaus steht eine höchst variierende Paketgröße von 64 Bytes bis 1500 Bytes zur Verfügung. Switches ermöglichen eine effiziente Ressourcenausnutzung bei der Gestaltung von Topologien. Darüber hinaus bietet Ethernet eine Vielzahl von Erweiterungen wie virtuelle Netzwerke oder die Vergabe von Prioritäten um *Quality of Service* (QoS) zu unterstützen. Die Erweiterung von Ethernet um *Audio Video Bridging* (AVB) [IEE12a] bietet zudem die Möglichkeiten, Datenraten zwischen Kommunikationsverbindungen zu reservieren, Zeitsynchronisation zwischen Endpunkten anzubieten und zeitsensitive Daten durch bestimmte Sendeeigenschaften in den Knoten optimiert zu übertragen. Das

IP Protokoll und höher angesiedelte Transportprotokolle wie UDP, TCP oder IEEE 1722 [IEE11a] bieten zudem weitere Dienste. Beispielweise ermöglicht TCP zuverlässige Kommunikation zwischen zwei Endpunkten und IEEE 1722 bietet ein standardisiertes Rahmenformat für die Übertragung von Video- und Audiosignalen, welches beispielsweise zur Anbindung von Kameras genutzt werden kann. In Richtung der Applikation existieren weitere *Middleware* Lösungen zur Bereitstellung von entfernten Prozeduraufrufen [Req76] oder zur Nutzung von *Service Discovery* [CAG05, CK11b, CK11a]. Entfernte Prozeduraufrufe werden dazu benutzt, um Interprozess-Kommunikation zu abzubilden. Somit wird es möglich, entfernte Prozeduraufrufe (üblicherweise befindet sich die Funktion auf einem anderen Steuergeräten im Netzwerk) zu tätigen, ohne sich im gleichen Adressraum befinden zu müssen und ohne explizit auf die Serialisierungsmechanismen der unterschiedlichen Partner Rücksicht nehmen zu müssen. Service Discovery wird unter anderem dafür eingesetzt, um Services im Netzwerk anzukündigen. Software-Komponenten können sich dann dynamisch für bestimmte Services registrieren oder deren Anmeldung widerrufen.

Ein guter Überblick über den Stand der Forschungs- und Entwicklungsäktivitäten zum Thema Ethernet und IP im Automobilbereich wurde auf [Mat11] vorgestellt. Mehrere Automobilhersteller [Fri11, JO11, SBLS11, Kri11, Got11] präsentierten ihre aktuellen Forschungsaktivitäten. Zukünftige Themen sind hier vor allem die Nutzung von Ethernet zur Vernetzung von Kameras und der Einsatz von Ethernet als Hochgeschwindigkeitsnetzwerk für die Bildung alternativer Bordnetzarchitekturen. Des Weiteren stellten diverse Zulieferer [Sin11, Nöb11, Kre11, Pow11, Hoe11] ihre Forschungsaktivitäten vor. Schwerpunkt war hier das applikative Szenario der Übertragung von Echtzeit-Video- und Audio über Ethernet und Ethernet AVB. Zudem wurden die Themen EMV-Stabilität von Ethernet und ein Vorschlag für eine Middleware Lösung vorgestellt, welche auf die Automobilen Einsatzszenarien zugeschnitten ist [KKM$^+$11, Völ11].

Die eigenen Beiträge dieser Dissertation sind in die drei Bereiche 1.) *Grundlegende Technologieabsicherung von Ethernet/IP-basierten Kommunikationsnetzwerken im Automobil*, 2.) *Integrations- und Migrationskonzepte für Ethernet/IP-basierte E/E-Architekturen* und 3.) *Simulation und Testen von Ethernet/IP-basierten E/E-Architekturen* gegliedert.

Grundlegende Technologieabsicherung von Ethernet/IP-basierten Kommunikationsnetzwerken im Automobil

Um eine neue Kommunikationstechnologie im Fahrzeug einsetzen zu können, ist es notwendig eine Mehrzahl von grundlegenden Anforderungen an ein Kommunikationssystem im Fahrzeug zu erfüllen. Diese Dissertation befasst sich mit Ermittlung von Bit und Restfehlerraten, Paket-basierter Synchronisationsgenauigkeit von verteilten Knoten und der Optimierung der Sendedatenrate eines Ethernet-Knotens.

Analyse von Bit- und Restfehlerraten Als Restfehler im Bezug auf Kommunikationssysteme werden Fehler in Datenpaketen bezeichnet, welche nicht durch Fehlererkennungsmechanismen erkannt werden. Daraus folgt, dass ein fehlerhaftes Paket an die empfangene Applikation weitergeleitet wird und dies wiederrum kann das Verhalten einer Applikation negativ beeinflussen. Methoden zur Bestimmung der Restfehlerrate sind 1.) die *direkte Code-Analyse*, 2.) die *transformierte Code-Analyse*, 3.) der *stochastische Automat* und die *Monte-Carolo Simulation*. Die analytischen Ansätze sind jedoch nur für kleine Paketgrößen effizient berechenbar und die Ergebnisse der simulativen Ansätze hängen von der Anzahl der simulierten Stichproben im Vergleich zur Anzahl der undetektierbaren Fehler ab. Eine wesentliche Eingangsgröße für alle Algorithmen ist die Häufigkeit und Verteilung von Bitfehlern. Ohne diese Eingangsgröße, ist eine Berechnung der Restfehlerwahrscheinlichkeit nicht möglich. Aus diesem Grund wird in dieser Dissertation ein Konzept entwickelt und implementiert, um das Auftreten von Bitfehlern für eine bestehende Verbindung zu bestimmen. Zudem werden die Restfehlerraten in Abhängigkeit der eingesetzten Fehlererkennungsmechanismen von Ethernet, IP und UDP ermittelt. Für kritische Verlegewege werden Fahrzeugmessungen durchgeführt, um isolierte Bitfehler und deren Verteilungen aufzeigen. Darüber hinaus wird dargestellt, dass während der gesamten Messungen kein einziger Restfehler aufgetreten ist.

Genauigkeit von Paket-basierten Synchronisationsmechanismen Heutige Kommunikationstechnologien basieren in der Regel auf einem physikalisch geteilten Medium. In solch einem Fall basiert eine Zeitsynchronisation von Endknoten auf dem gleichzeitigen Empfang einer Bitmaske oder eines Pakets. Im Falle von

geswitchten Ethernet-Netzwerken kommen Paket-basierte Synchronisationsmechanismen zum Einsatz, um eine globale Zeitbasis zu etablieren. Typische Algorithmen sind IEEE 1588-2002 [IEE02], IEEE 1588-2008 [IEE08a] und IEEE 802.1AS [IEE10b]. Die Genauigkeit dieser Algorithmen hängt stark von der Genauigkeit der eingesetzten Oszillatoren ab. Bis heute wurden diese Algorithmen überwiegend in Umgebungen wie Rechenzentren oder PCs eingesetzt, wo stabile Temperaturbedingungen vorherrschen. Um die Einsetzbarkeit solcher Algorithmen in Fahrzeugumgebungen sicherstellen zu können, wird in dieser Dissertation die Synchronisationsgenauigkeit für IEEE 802.1AS für verändernde Temperaturbedingungen evaluiert. Hierzu wird ein IEEE 802.1AS Netzwerk eingerichtet, um die Synchronisationsgenauigkeit für unterschiedliche Temperaturbedingungen messen zu können. Es wird aufgezeigt, dass der eingesetzte Paket-basierte Algorithmus mit Temperaturveränderungen zurecht kommt. Sogar beim Einsatz von Schock-Klimaschränke zur raschen Änderung der Umgebungstemperatur. Die Ergebnisse der Temperaturmessungen wurden in [KZS+11•] veröffentlicht.

Datendurchsatz Optimierung von Ressourcen-beschränkten Systemen
Datendurchsatzmessungen vorgestellt in [Ker08•] zeigen auf, dass in Ressourcenbeschränkten eingebetteten Ethernet-basierten Systemen der messbare Datendurchsatz deutlich unter der theoretisch möglichen oberen Grenze des darunterliegenden physikalischen Schicht ist. Sogar durch den Einsatz von optimierten Speicherkopiermechanismen wie *Direct Memory Access* (DMA) ist die erreichbare Datenrate limitiert. Der Hauptgrund hierfür ist der ressourcenhungrige Netzwerk-Stack für die Ausführung der Protokolle der höheren Schichten, welche in Software ausgeführt werden. Diese Dissertation präsentiert unterschiedliche Konzepte um die Sendedatenrate einer Ethernet und IP Verbindung zu optimieren. Die Konzepte bewegen sich von Anpassungen in Software bis hin zu Hardware-optimierten Ausprägungen. Es wird aufgezeigt, dass die Implementierungen der einzelnen Konzepte Datenraten bis hin zur vollen Auslastung hin unterstützten. Dies wird durch die Deaktivierung von nicht genutzten Funktionen und die Einbringung von zusätzlicher Hardware ermöglicht. Die unterschiedlichen Optimierungskonzepte wurden in [KSS+10•] veröffentlicht.

Integrations- und Migrationskonzepte für Ethernet/IP-basierte E/E-Architekturen

Durch den Einsatz einer neuen Kommunikationstechnologie wie Ethernet und IP werden zwingend Gateway-Konzept benötigt, um Kommunikation ins Restfahrzeugnetz ermöglichen zu können. Zudem können auf lange Sicht sogar Ersetzungsstrategien zum Einsatz kommen, welche heutige Kommunikationssysteme durch Ethernet und IP ersetzen. Die Beiträge, um ein bestehendes Fahrzeugnetz mit Ethernet und IP verbinden zu können und die Möglichkeit der Ersetzung von etablierten Kommunikationssystemen werden im Folgenden vorgestellt.

Gateway Strategien Heutige Bordnetzarchitekturen basieren typischerweise auf einem Ansatz mit einem zentralen Gateway, welches die unterschiedlichen Kommunikationstechnologien miteinander verbindet. Hierzu wird ein Gateway mit einer Mehrzahl an Kommunikationsschnittstellen ausgestattet, welche dann den Datenaustausch zwischen den unterschiedlichen Netzwerken ermöglichen. Durch die Hinzunahme von Ethernet und IP als eine Schnittstelle werden Konzepte benötigt, welche die Abbildung der Informationen zwischen den einzelnen Kommunikationstechnologien beschreibt. Diese Dissertation beschreibt hierzu Konzepte, um Datenverkehr sowohl zwischen CAN und Ethernet/IP als auch zwischen FlexRay und Ethernet AVB weiterleiten zu können. Hierbei werden unterschiedliche Paketierungsmechanismen von einfachen Eins-zu-Eins Abbildungen bis hin zu Viele-zu-Eins Abbildungen entwickelt, implementiert und evaluiert. Die prototypische CAN/Ethernet-Gateway Umsetzung wird zudem in einer aktuellen E-Klasse verbaut und getestet. Im Fall von FlexRay kommen Ethernet AVB Mechanismen zum Einsatz, um definierte Datenraten und zeitliche Anforderungen zwischen den Endpunkten garantieren zu können. Hierbei werden FlexRay-Pakete in IEEE 1722 Pakete paketiert. Ein IEEE 1722 Paketvorschlag wurde hierzu vorgestellt [KSG+11●] und die Ergebnisse der Gateway Untersuchungen wurden in [KRST11●] veröffentlicht.

Ersetzungsstrategien Neben der Kommunikation zwischen existierenden Technologien und Ethernet/IP stellt die längerfristige partielle oder komplette Ersetzung von bestehenden Technologien mit Ethernet und IP einen weiteren attraktiven Anwendungsfall dar. Ein typischer Fall ist die Ersetzung von CAN durch Ethernet, falls

eine Ethernet-Verbindung im gleichen Verbauraum verlegt ist. Diese Dissertation präsentiert zwei Konzepte für die Ersetzung von CAN oder FlexRay durch Ethernet und IP. Die zu untersuchenden Hauptunterschiede sind hierbei die Adressierung, das Sendeverhalten und die Paketgröße der unterschiedlichen Technologien. Im Fall von CAN wird eine Referenzumsetzung bestehend aus einer Eins-zu-Eins Abbildung mit einem evolutionären und einem Regel-basierten Ansatz verglichen, um die Abbildung der CAN-Nachrichten in Ethernet-Pakete zu optimieren. Alle Ansätze funktionierten hierbei automatisiert und werden für reelle Kommunikationsbeschreibungen angewendet. Im Fall von FlexRay kommen virtuelle Netzwerke zum Einsatz, um die Adressierungsproblematik zu lösen. Bei der Paketierung wird zwischen einem Ansatz basierend auf einem FlexRay-Zyklus und einem Ansatz über mehreren Zyklen hinweg unterschieden. Kommunikation über mehrere Zyklen hinweg wird typischerweise für Transportprotokolle genutzt, um große Mengen von Daten übertragen zu können. Für beide Technologien wird aufgezeigt, dass eine Ersetzung durch Ethernet und IP möglich ist. Zudem werden detaillierte Vergleiche zwischen den Ansätzen vorgestellt. Zielgrößen sind hierbei die durchschnittliche Anzahl an Nachrichten pro Paket, das Verhältnis zwischen Paketkopf und Paketnutzdaten oder die notwendigen Datenraten im Ethernet-Verbund. Hierbei kommen wiederrum reale Netzwerkbeschreibungen zum Einsatz. Die Ergebnisse wurden in [KST11●] veröffentlicht.

Simulation und Testen von Ethernet/IP-basierten E/E-Architekturen

Kapitel 4 dieser Dissertation umfasst die Simulation, Restbussimulation und das Überwachen von Punkt-zu-Punkt Ethernet-Netzwerken sowie die Ausführung von Stresstests. Heutige automobilspezifische Testtools basieren fast ausschließlich auf einem physikalisch-geteilten Medium. In solch einem Fall ist das Aufzeichnen oder Generieren von Nachrichten eine einfache Aufgabe, da hier nur der Testknoten zum Netzwerk hinzugefügt werden muss. In Punkt-zu-Punkt Netzwerken mit Switches werden neue Ansätze für die Simulation, Paketgenerierung und Paketaufzeichnung benötigt, da nicht der komplette Datenverkehr eines Netzwerks auf jeder Verbindung sichtbar ist. Diese Dissertation präsentiert ein Ethernet-Test-Switch Konzept, welches das Testen von Punkt-zu-Punkt Netzwerken im Fahrzeug erlaubt. Zusätzlich wird eine Schnittstelle zur Restbussimulation vorgestellt. Diese ermöglicht die Anbindung ei-

ner Simulationsumgebung an den Ethernet-Test-Switch. Somit lassen sich Systeme, welche nur teilweise in Hardware verfügbar sind, simulieren.

Simulations von Punkt-zu-Punkt Netzwerken Simulation ist eine gängige Methode zur Evaluierung von System im frühen Entwicklungsprozess. Hierdurch wird es möglich ein Verständnis für das Verhalten eines Protokolls oder die zeitlichen Abläufe eines Systems zu erhalten ohne je Hardware vorliegen zu haben. Diese Dissertation beschreibt ein Konzept zur Simulation von Ethernet/IP-basierten Systemen inklusive Software-Laufzeiten und Datenabhängigkeiten zwischen Funktionsblöcken mit Hilfe eines ereignisgesteuerten Simulators. Alle notwendigen Erweiterungen zur Simulation von Ethernet und IP werden vorgestellt und in die Simulationsumgebung integriert. Eine Validierung der Implementierung wird anhand verschiedener Fallbeispiele durchgeführt. Es wird gezeigt, dass die ereignisgesteuerte Simulation eine leistungsfähige Methodik ist, um das funktionale und zeitliche Verhalten eines Ethernet/IP-basierten Systems zu simulieren. Die Evaluierungsergebnisse eines Multi-Kamera-Systems wurden in [RKH+10•] veröffentlicht.

Restbussimulation von Punkt-zu-Punkt Ethernet-Netwerken Wenn die erste Hardware während der Entwicklungsphase des zu Zielsystems zur Verfügung steht, bekommt das Testen einen grundlegenden Platz im Entwicklungsprozess um die Verlässlichkeit des Systems sicherzustellen. Dabei wird zwischen Systemen unterschieden welche komplett in Hardware und Software vorliegen und Systeme welche Restbussimulation benötigen, da nur bestimmte Teile des Systems in Hardware und Software vorliegen. Für diesen Bereich wird ein Ethernet-Test-Switch Konzept vorgestellt und implementiert. Dieses erlaubt das Einbringen von zusätzlichem Datenverkehr um Stresstests zu erzeugen, das Mitlesen des gesamten Datenverkehrs durch den Switch und die Unterstützung von Restbussimulation zur Untersuchung von Systemen, welche teilweise in Hardware vorliegen. Das Konzept, dessen Implementierung und die Leistungsuntersuchung des Ethernet-Test-Switches wurde in [KZST11•] veröffentlicht. Darüber hinaus wurde das zugrunde liegende Konzept in [KSZ11•] patentiert.

Die Ergebnisse der Kapitel 1.) *Grundlegende Technologieabsicherung von Ethernet/IP-basierten Kommunikationsnetzwerken im Automobil*, 2.) *Integrations- und Migrationskonzepte für Ethernet/IP-basierte E/E-Architekturen* und 3.) *Simulation und Testen von Ethernet/IP-basierten E/E-Architekturen* sind wie folgt:

Beiträge zur grundlegenden Technologieabsicherung von Ethernet/IP-basierten Kommunikationsnetzwerken im Automobil Um Ethernet als eine zusätzliche Kommunikationstechnologie für die Vernetzung innerhalb eines Fahrzeugs einsetzen zu können, müssen grundlegende Anforderungen im Automobilbereich erfüllt werden. Kapitel 2 beschrieb hierzu Konzepte 1.) zur Bestimmung der Bit- und Restfehlerrate von Ethernet-Verbindungen, 2.) zur Messung der Synchronisationsgenauigkeit von Paket-basierten Synchronisationsalgorithmen und 3.) zur Optimierung der Sendedatenrate beim Einsatz von günstigen Ressourcen-beschränkten Prozessoren. Im Bereich der Fehlerratenanalyse wurde ein Konzept zur Ermittlung der Fehlerraten für Ethernet-PHYs entwickelt. Das Ziel war hierbei die Bestimmung der Bit-, Paket- und Restfehlerraten für unterschiedliche Fehlererkennungsmechanismen. Darüber hinaus wurden die Bitfehlerverteilung und die Anzahl der Bitfehler pro Paket bestimmt. Die durchgeführten BCI- und Fahrzeugmessungen zeigten neuartige Bit- und Paketfehlerergebnisse für gegebene Ethernet-PHY Umsetzungen. Um die erhöhten Temperaturanforderungen im Automobilbereich erfüllen zu können wurde ein Konzept zur Bestimmung der Synchronisationsgenauigkeit für variierende Temperaturbedingungen vorgestellt. Für die Messungen kamen Industrieklimaschränke zum Einsatz. Die Synchronisation der Endknoten wurde mittels IEEE 802.1AS und Ethernet als Kommunikationstechnologie durchgeführt. Über mehrere Tage hinweg ergaben gründliche Messungen eine beachtliche Menge von Ergebnissen welche aufzeigten, dass IEEE 802.1AS mit veränderlichen Temperaturbedingungen keine Probleme aufweist. Der Temperaturbereich für die Messungen bewegte sich zwischen -10 °C und $+70$ °C. Abschließend wurden Messungen in Schock-Klimaschränken durchgeführt, welche eine rapide Temperaturveränderung von 80 Kelvin in weniger als 5 Sekunden ermöglichten. In Summe konnte somit gezeigt werden, dass Ethernet in der Lage ist synchronisierte Uhren im Nanosekundenbereich anzubieten. Des Weiteren wurde ein Konzept vorgestellt, welches die Sendedatenrate eines Ethernet-fähigen Steuergerätes optimierte. Unterschiedliche Varianten nutzen Softwareanpassungen und zusätzliche Hardwareunterstützung zur Steigerung der Sendeleistung. Es konnte aufgezeigt werden, dass ein Ethernet-fähiges Steuergerät mit einem 85 MHz FPGA-Prozessor eine Datenrate von knapp 1 GBit/s unterstützen kann.

Beiträge zu Integrations- und Migrationskonzepten für Ethernet/IP-basierte E/E-Architekturen Aufgrund der positiven Ergebnisse der Technologieana-

lyse wurden in Kapitel 3 Gateway- und Ersetzungskonzepte entwickelt, um die Kommunikation mit existierenden Technologien wie CAN und FlexRay zu ermöglichen. Im Bereich der Gateways wurde zunächst ein CAN/Ethernet-Gateway Konzept entwickelt, prototypisch implementiert und anschließend mit realen Fahrzeugdaten und durch den Verbau in einer aktuellen E-Klasse getestet. Neben der technischen Umsetzbarkeit wurde somit eine detaillierte Evaluierung des Gateways durchgeführt. Es konnte aufgezeigt werden, dass die durchschnittliche Ende-zu-Ende Latenz über zwei Gateways zwischen 500 µs bei der Verwendung einer Eins-zu-Eins Abbildung und 4500 µs bei der Verwendung von gepufferten Ansätzen variierte. Die durchschnittliche Ethernet-Datenrate war sechs Mal höher als die CAN-Datenrate ohne Verwendung von Puffern. Mit Pufferung konnte die Ethernet-Datenrate auf die doppelte CAN-Datenrate gesenkt werden. Die maximale Auslastung des Gateway-Prozessors lag ungefähr bei 27%. Im nächsten Schritt wurde ein FlexRay/AVB-Gateway Konzept erarbeitet, um Daten des statischen Segments von FlexRay nach Ethernet AVB und umgekehrt weiterzuleiten. Das Umsetzungskonzept unterschied zwischen Abbildungen, welches pro Zyklus erstellt wurde und Abbildungen, welche mehrere Zyklen mit einbezogen. Dies wurde für die Ersetzung von typischen Transportprotokollen genutzt. Auf der Basis von realen Kommunikatoinmatrizen konnte somit gezeigt werden, dass die Datenrate im AVB-Netzwerk durchschnittlich drei mal höher als die FlexRay-Datenrate ist. Im zweiten Abschnitt des Kapitels wurden dann Ersetzungsstrategien und -konzepte für CAN und FlexRay entwickelt und ausgewertet. Für die CAN-Ersetzung wurde eine Eins-zu-Eins-Referenzabbildung, eine Abbildung basierend auf einem evolutionären Algorithmus und eine Regel-basierte Abbildung generiert. Die Ergebnisse zeigten den ungünstigsten Fall unter der Annahme, dass spontane Nachrichten als zyklische Nachrichten abgebildet werden. In diesem Fall war die ungünstigste Datenrate 100-mal höher als die zugrunde liegende CAN-Datenrate. Der evolutionäre und Regel-basierte Ansatz konnte die Datenrate im Ethernet halbieren. Im Fall von FlexRay wurde das zugrunde liegende FlexRay-Gateway Konzept adaptiert. Hierzu wurden unterschiedliche Verbindungen zwischen Steuergeräten auf unterschiedliche virtuelle Netze abgebildet. Ein Testaufbau zeigte eine dreifach höhere AVB-Datenrate im Vergleich zur FlexRay-Datenrate. Durch Pufferung konnte die Datenrate fast auf die zugrunde liegende FlexRay-Datenrate reduziert werden. Darüber hinaus war die durchschnittlich gemessene Latenz vergleichbar zur Latenz, welche im FlexRay-Netzwerk aufgetreten ist. Zusammenfassend wurden in diesem

Kapitel vier Konzepte für die Verbindung und Ersetzung von etablierten Kommunikationstechnologien mit Ethernet und IP erarbeitet. Prototypische Implementierungen bestätigten die technische Umsetzbarkeit, welche sogar in einem aktuellen E-Klasse Fahrzeug gezeigt wurde.

Beiträge zu Simulation und Testen von Ethernet/IP-basierten E/E-Architekturen Neben den Konzepten zur Umsetzung unterschiedlicher Infrastrukturelementen, spielt das Testen von Ethernet-basierten Systemen eine wichtige Rolle im Entwicklungsprozess, um die Verfügbarkeit und Korrektheit unterschiedlicher integrierter Applikationen garantieren zu können. Um ein mit Ethernet-vernetztes System bereits in einer frühen Entwicklungsphase – wenn keine Hardware und Software zur Verfügung steht – evaluieren zu können, wurde ein Simulationskonzept erarbeitet. Darüber hinaus wurde eine Konzept erstellt, um eine Simulation von Systemen zu ermöglichen, welche bereits teilweise zur Verfügung stehen. Im Bereich der Simulation wurde hierzu ein diskreter ereignisgesteuerter Simulator adaptiert und um die Module Ethernet/IP und die Möglichkeit zur Simulation von Tasks und deren Abhängigkeiten für unterschiedliche Scheduling-Mechanismen erweitert. Hierfür wurden Ethernet-Tranceiver, -Controller und Switches als eigenständige Simulationsmodule modelliert. Zusätzlich wurde ein Task-Modell eingeführt, um den Software-Netzwerkstack simulieren zu können. Auf dieser Grundlage wurde anschließend ein Multi-Kamerasystem evaluiert. Durch die Simulation konnte aufgezeigt werden, dass die gewählte Netzwerkarchitektur in der Lage ist die aufkommenden Netzwerkpakete zu transportieren, solange die Paketgröße größer als 600 Bytes ist. Kürzere Pakete verursachten einen Ressourcenkonflikt innerhalb des zentralen Steuergerätes im Prozessor, da dieser im Software-Netzwerkstack mehr Pakete pro Zeit verarbeiten musste. Es konnte somit gezeigt werden, dass in diesem Falle ein leistunsfähigerer Prozessor innerhalb des zentralen Steuergerätes gewählt werden müsste. Zusätzlich wurde die Ende-zu-Ende-Latenz ermittelt und der Anteil der Netzwerk-Latenz zur gesamten Latenz bestimmt. Das zweite Konzept dieses Kapitels befasste sich mit dem Mitlesen und Einfügen von Paketen in ein bestehendes Netzwerk sowie der Restbussimulation von geswitchten Ethernet-Netzwerken. Hierbei wurde ein Konzept für einen transparenten Ethernet-Switch entwickelt, welcher durch eine transparente “Switch-Fabric” und zusätzliche Module in der Lage ist, den internen Switch-Datenverkehr über eine externe Schnittstelle an ein entferntes Evaluierungstool weiterzugeben. Somit wa-

re es möglich, den kompletten Datenverkehr durch einen Switch mitzulesen ohne auf Techniken wie Port-Spiegelung zurück greifen zu müssen. Zusätzlich wurden hochpräzise Zeitstempel eingeführt, um das tatsächliche Verhalten des Netzwerks aus zeitlicher Sicht im Evaluierungstool rekonstruieren zu können. Ein Paketgenerator ermöglichte das Einbringen von zusätzlichen Datenpaketen in das zu testende Netzwerk mit unterschiedlichen Größen und definierten Häufigkeiten. Letztlich wurde die externe Schnittstelle erweitert, um eine Verbindung zu einer Simulationsumgebung zu schaffen damit beispielsweise eine Restbussimulation ermöglicht werden kann. Beispielweise könnte durch eine Verbindung des oben vorgestellten Simulators mit dem Ethernet-Test-Switch eine Restbussimulation durchgeführt werden. Die Vermessung des Switches selbst ergab eine maximale Datenrate von 4.000 MBit/s für das Mitlesen von Paketen durch den Switch. Die Datenrate wurde hierbei durch die Schnittstelle zum Evaluierungsrechner begrenzt. Die Paketgenerierung arbeitete bis zur maximalen Datenrate und für die Restbussimulation konnten 4 MBit/s erreicht werden.

Zukünftige Forschungsthemen finden sich unter anderem in den drei Bereiche 1.) *Tooling*, 2.) *Protokolle höherer Schichten* und 3.) *Timing-Analyse*. Im Bereich Tooling gilt, es neue Dateiformate zu entwickeln, welche in der Lage sind, die Ethernet-Topologie und die IP-basierte Kommunikation zwischen Steuergeräten eines Netzwerks zu beschreiben. Neben den Schnittstellenbeschreibungen auf Steuergeräteebene beschreiben neue Systemspezifikationen, wie beispielweise AUTOSAR, außerdem die Kommunikation auf Softwarekomponentenebene. Somit muss zusätzlich die Abbildung der Schnittstellen auf Steuergeräte- und Softwarekomponentenebene mit berücksichtigt werden. Des Weiteren müssen die Gateway-Beschreibungstabellen für Ethernet/IP-basierte Kommunikation erweitert werden. Ziel sollte hier die Definition einer Menge von Formaten sein, welche Automobilhersteller-übergreifend zum Einsatz kommen. Ein weiterer wichtiger Punkt ist das Verständnis der höheren Protokolle und deren Verhalten bei Ressourcen-beschränkten Prozessoren und Speichern. Heutige Standard-Netzwerkstacks nutzen beispielsweise eigene Speicherbereiche und Administrationstabellen für deren Verwaltung. Ein weiterer Punkt ist die Untersuchung der Wechselwirkungen zwischen unterschiedlichen Protokollen. Hier ist sicherzustellen, dass die unterschiedlichen Protokolle sich nicht gegenseitig beeinflussen und es zu keinen Konflikten kommt. Durch den Einsatz von zeitkritischen oder Echtzeitfähigen Applikationen im Ethernet-Verbund, bekommt die Timinganalyse solcher

Netzwerke eine wichtige Rolle beim Entwurf eines Netzwerks. Typische Regelsysteme fordern in der Regel definierte Höchstgrenzen für die Übermittlungszeit innerhalb des Netzwerks. Somit ist sicherzustellen, dass weder Paketverlust noch eine Überschreitung der maximalen Übertragungsdauer auftritt. Einen detaillierten Überblick über die zeitliche Analyse von bestehenden Kommunikationstechnologien ist in [ST12] beschrieben.

Bibliography

[Alt09] Altera Corporation. *Nios II Development Kit, Stratix II Edition*. Altera Cooperation, San Jose, USA, September 2009. http://www.altera.com/products/devkits/altera/kit-niosii-2S60.html/.

[Alt10] Altera Corporation. *Acceleration Nios II Networking Applications*. Application Note – AN-440-2.0, August 2010.

[ASA11] ASAM MCD-2 NET. *FIBEX 4.0 Standard*. ASAM e.V., Höhenkirchen, Germany, September 2011.

[AUT08] AUTOSAR. *Requirements on Gateway*. AUTOSAR Development Cooperation, Munich, Germany, February 2008.

[AUT11] AUTOSAR Standard 063. *AUTOSAR System Template 4.2.0*. AUTOSAR Development Cooperation, Munich, Germany, November 2011.

[AUT12] AUTOSAR – AUTomotive Open System ARchitecture, October 2012. http://www.autosar.org/.

[BGH+10] Roland Bless, Gerrit Grotewold, Christian Haas, Boris Hackstein, Stefan Hofmann, Anke Jentzsch, Alexander Kiening, Christoph Krauß, Julian Lamberty, Michael Müter, Peter Schoo, Lars Völker, and Christoph Werle. *A Security Model for Future Vehicular Electronic Infrastructures*. In Proceedings of 8th Embedded Security in Cars Conference (ESCAR2010), Bremen, Germany, November 2010.

[BGNV09] Jeff Burch, Kenneth Green, John Nakulsko, and Dieter Vook. *Verifying the Performance of Transparent Clocks in PTP Systems*. In Proceedings

of the International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS2009), Brecia, Italy, pages 1 – 7, October 2009.

[BH07] Josef Börcsök and H.-T. Hannen. *Determination of Bit Error and Residual Error Rates for Safety Critical Comunication*. In Proceedings of the 2nd International Conference on Systems (ICONS2007), Sainte-Luce, Martinique, April 2007.

[BRR11] Manfred Broy, Günter Reichart, and Lutz Rothhardt. *Architekturen softwarebasierter Funktionen im Fahrzeug: von den Anforderungen zur Umsetzung*. Informatik Spektrum, Volume 34, No. 1, pages 42 – 59, January 2011.

[CAG05] Stuart Cheshire, Bernard Aboba, and Eric Guttman. *Request for Comments 3927 – Dynamic Configuration of IPv4 Link-Local Addresses*. The Internet Engineering Task Force (IETF), Fremont, USA, May 2005.

[CAN91] CAN Specification. *CAN Specification – Version 2.0*. Robert Bosch GmbH, Stuttgart, Germany, 1991.

[Car12] Car 2 Car Communication Consortium, April 2012. http://www.car-to-car.org/.

[CGY01] Jeffrey S. Chase, Andrew J. Gallatin, and Kenneth G. Yocum. *End-System Optimizations for High-Speed TCP*. In IEEE Communications Magazine 4/2001, April 2001.

[Cha94] Joachim Charzinski. *Performance of the Error Detection Mechanism in CAN*. In Proceedings of the 1st International CAN Conference, Mainz, Germany, 1994.

[CK11a] Stuart Cheshire and Marc Krochmal. *Internet Draft Expires June 11, 2012 – DNS-Based Service Discovery*. The Internet Engineering Task Force (IETF), Fremont, USA, December 2011.

[CK11b] Stuart Cheshire and Marc Krochmal. *Internet Draft Expires June 11, 2012 – Multicast DNS*. The Internet Engineering Task Force (IETF), Fremont, USA, December 2011.

[CL07] Jonathan Chao and Bin Liu. *High Performance Switches and Routers*. John Wiley & Sons, Hoboken, USA, 1st edition, 2007.

[CL08] Christos Cassandras and Stéphane Lafortune. *Introduction to Discrete Event Systems*. Springer Science+Business Media, New York, USA, 2nd edition, 2008.

[EFH+08] John Eidson, Andrew Fernandez, Bruce Hamilton, Jad Naous, and Dieter Vook. *Spider Transparent Clock*. In Proceedings of the IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS2008), Ann Arbor, USA, pages 7 – 11, September 2008.

[Eth12a] EtherCAT, April 2012. http://www.beckhoff.de/EtherCAT/.

[Eth12b] Ethernet History, May 2012. http://timeline.ethernethistory.com/.

[Eth12c] Ethernet Powerlink, April 2012. http://www.ethernet-powerlink.com/.

[FFM+07] Paolo Ferrari, Alessandra Flamini, Daniele Maroli, Stefano Rinaldi, and Andrea Taroni. *Synchronization of the Probes of a Distributed Instrument for Real-Time Ethernet Networks*. In Proceedings of the IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS2007), Vienna, Austria, pages 33 – 40, October 2007.

[Fle10] FlexRay Consortium. *FlexRay Communication System Protocol Specification – Version 3.0.1*. FlexRay Consortium – Altran GmbH & Co. KG, Frankfurt am Main, Germany, October 2010.

[Fri11] Elmar Frickenstein. *Ethernet&IP in Cars – An Idea Whose Time Has Come.* In Proceedings of 1st Ethernet&IP@Automotive Day, Munich, Germany, November 2011.

[GGT09] Geoffrey Garner, Aaron Gelter, and Michael Johas Teener. *New Simulation and Test Results for IEEE 802.1AS Timing Performance*. In Proceedings of the International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS2009), Brecia, Italy, pages 1 – 7, October 2009.

[Got11] Hideki Goto. *Our Approach to Next-Generation Automotive Network*. In Proceedings of 1st Ethernet&IP@Automotive Day, Munich, Germany, November 2011.

[HH09] John Heidemann and Tom Henderson. *Network Simulator 2 Release NS-2.34*, June 2009. http://www.isi.edu/nsnam/ns/.

[HJ10] Jiho Han and Deog-Kyoon Jeong. *A Practical Implementation of IEEE 1588-2008 Transparent Clock for Distributed Measurement and Control Systems*. In IEEE Transactions on Instrumentation and Measurement (TIM2010), February 2010.

[HK11] Stefan Hofmann and Rudolf Kasseckert. *Towards a Security Architecture for IP-based Optical Transmission Systems*. In Bell Labs Technical Journal – Volume 16 – Issue 1, June 2011.

[Hoe11] Rob Hoeben. *Automotive Ethernet Strategy*. In Proceedings of 1st Ethernet&IP@Automotive Day, Munich, Germany, November 2011.

[IEE02] IEEE 1588-2002. *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. The Institute of Electrical and Electronics Engineers, Inc., New York, USA, November 2002.

[IEE06] IEEE 802.1Q-2005. *IEEE Standard for Local and Metropolitan Area Network – Virtual Bridged Local Area Networks*. The Institute of Electrical and Electronics Engineers, Inc., New York, USA, May 2006.

[IEE08a] IEEE 1588-2008. *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. The Institute of Electrical and Electronics Engineers, Inc., New York, USA, July 2008.

[IEE08b] IEEE 802.3. *IEEE Standard for Local and Metropolitan Area Network – Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Networks – Part3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*. The Institute of Electrical and Electronics Engineers, Inc., New York, USA, December 2008.

[IEE10a] IEEE 1901-2010. *IEEE Standard for Broadband over Power Line Networks: Medium Access Control and Physical Layer Specifications*. The Institute of Electrical and Electronics Engineers, Inc., New York, USA, December 2010.

[IEE10b] IEEE 802.1AS/D7.6. *Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*. The Institute of Electrical and Electronics Engineers, Inc., New York, USA, November 2010.

[IEE10c] IEEE 802.1Qav. *Virtual Bridged Local Area Networks – Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams*. The Institute of Electrical and Electronics Engineers, Inc., New York, USA, January 2010.

[IEE10d] IEEE 802.1Qav. *Virtual Bridged Local Area Networks – Amendment 14: Stream Reservation Protocol (SRP)*. The Institute of Electrical and Electronics Engineers, Inc., New York, USA, September 2010.

[IEE11a] IEEE 1722. *IEEE Standard for Layer 2 Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks*. The Institute of Electrical and Electronics Engineers, Inc., New York, USA, May 2011.

[IEE11b] IEEE 802.1Q-2011. *Media Access Control (MAC) Bridges and Virtual Bridge Local Area Networks*. The Institute of Electrical and Electronics Engineers, Inc., New York, USA, August 2011.

[IEE12a] IEEE 802.1 A/V Bridging Task Group, April 2012. http://www.ieee802.org/1/.

[IEE12b] IEEE 802.3 Working Group, April 2012. http://www.ieee802.org/3/.

[Ins04] Eddie Insam. *Interfacing 100T Ethernet and Embedded Systems*. In Circuit Cellar Magazine 11/2004, November 2004.

[ISO11a] ISO 13400-1. *Road Vehicles – Diagnostic Communication over Internet Protocol (DoIP) – Part 1: General Information and Use Case Definition*. International Organization for Standardization (OSI), Geneva, Switzerland, October 2011.

[ISO11b] ISO 15765-2. *Road Vehicles – Diagnostic Communication over Controller Area Network (DoCAN) – Part 2: Transport Protocol and Network Layer Services*. International Organization for Standardization (OSI), Geneva, Switzerland, November 2011.

[ISO11c] ISO 26262-9. *Road vehicles – Functional safety – Part 9: Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses*, November 2011.

[ISO11d] ISO/CD 15118-2. *Road Vehicles – Vehicle to Grid Communication Interface – Part 2: Technical Protocol Description and Open System Interconnections (OSI) Requirements*. International Organization for Standardization (OSI), Geneva, Switzerland, July 2011.

[ISO12] ISO. International Organization for Standardization, http://www.iso.org/, 2012.

[Ixi11] Ixia. *Ixia – IxNetwork*, February 2011. http://www.ixiacom.com/.

[JO11] Markus Joachim and Kenneth Orlando. *Ethernet for Backbone and Control Applications*. In Proceedings of 1st Ethernet&IP@Automotive Day, Munich, Germany, November 2011.

[KÖ7] Daniel Köhler. *A Practical Implementation of an IEEE 1588 supporting Ethernet Switch*. In Proceedings of the IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS2007), Vienna, Austria, pages 134 – 137, October 2007.

[KC04] Philip Koopman and Tridib Chakravarty. *Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks*. In Proceedings of the International Conference on Dependable Systems and Networks (DSN2004), Florence, Italy, June 2004.

[KKM+11] Bernd Körber, Ronny Kunz, Norman Müller, Matthias Trebeck, and Tom Wunderlich. *Methodologies for EMC Optimization of Automotive Ethernet Systems*. In Proceedings of 1st Ethernet&IP@Automotive Day, Munich, Germany, November 2011.

[KKS+10] Christoph Krauß, Alexander Kiening, Peter Schoo, Roland Bless, Christoph Werle, and Christian Haas. *Towards Secure Integration of IP into Vehicles*. In Proceedings of Automotive Safety & Security, Stuttgart, Germany, January 2010.

[Koo02] Philip Koopman. *32-Bit Cyclic Redundancy Codes for Internet Applications*. In Proceedings of the International Conference on Dependable Systems and Networks (DSN2002), Bethesda, USA, June 2002.

[Kre11] Nick Kreifeldt. *AVB and the New Converged Backbone*. In Proceedings of 1st Ethernet&IP@Automotive Day, Munich, Germany, November 2011.

[Kri11] Olaf Krieger. *Ethernet and IP @ VW*. In Proceedings of 1st Ethernet&IP@Automotive Day, Munich, Germany, November 2011.

[LGRT11] Martin Lukasiewycz, Michael Glaß, Felix Reimann, and Jürgen Teich. Opt4J: A Modular Framework for Meta-Heuristic Optimization. In *GECCO*, pages 1723–1730, 2011.

[LIN10] LIN Consortium. *LIN Specification Package – Revision 2.2*. LIN Administration – Altran GmbH & Co. KG, München, Germany, December 2010.

[LKVZ11] Hyuang-Taek Lim, Benjamin Krebs, Lars Völker, and Peter Zahrer. *Performance Evaluation of the Inter-Domain Communication in a Switched Ethernet Based In-Car Network*. In Proceedings of 36th

IEEE Conference on Local Computer Networks (LCN2011), Bonn, Germany, October 2011.

[LVD95] LVDS Interface Standard. *Low Voltage Differential Signaling Interface Standard – ANSI/TIA/EIA-644-1995*, 1995.

[LWH11] Hyuang-Taek Lim, Kay Weckemann, and Daniel Herrscher. *Performance Study of an In-Car Switched Ethernet Network without Prioritization*. In Proceedings of 3rd International Workshop on Communication Technologies for Vehicles (Nets4Cars2011), March 2011.

[Mat11] Kirsten Matheus. *1st Ethernet & IP Automotive Techday*. BMW, Munich, Germany, November 2011.

[Mer09] Mercedes Benz Company Standard. Electrical and electronic components in passenger cars up to 3.5t – general requirements, test conditions and test – part 2: Environmental requirements, November 2009.

[Met94] Robert Metcalfe. *How Ethernet Was Invented*. IEEE Annals of the History of Computing, Volume 16, No. 4, page 84 and following pages, 1994.

[MOS10] MOST Cooperation. *Media Oriented Systems Transport Specification – Revision 3.0 E2*. MOST Cooperation Administration , Frankfurt am Main, Germany, July 2010.

[MPSH07] Tina Mattes, Jörg Pfahler, Frank Schiller, and Thomas Honold. *Analysis of Combinations of CRC in Industrial Communication*. In Proceedings of the 26th International Conference on Computer Safety, Reliability, and Security (SAFECOMP2007), Nuremberg, Germany, September 2007.

[MSM+08] Tina Mattes, Frank Schiller, Annemarie Mörwald, Jörg Pfahler, and Thomas Honold. *Safety Proof of Combinations of CRC for Industrial Communication*. In the Journal of Applied Computer Science, Vol. 16 No. 1, pp. 15 – 32, 2008.

[Mur09] Richard Murphy. *A CAN to FlexRay Migration Framework*. Waterford Institute of Technology, Ireland, September 2009.

[Net11] NetOptics Inc. *NetOptics – Network Taps*, February 2011. http://www.netoptics.com/.

[Nöb11] Josef Nöbauer. *Migration from MOST and FlexRay Based Networks to Ethernet by Maintaining QoS*. In Proceedings of 1st Ethernet&IP@Automotive Day, Munich, Germany, November 2011.

[OSI12] OSI – International Organization for Standardization, September 2012. http://www.iso.org/.

[OV05] Alberto Ornaghi and Marco Valleri. *Ettercap Release NG-0.7.3*, May 2005. http://ettercap.sourceforge.net/.

[Pet62] Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Darmstadt University of Technology, Germany, 1962.

[Pfe01] Tilo Pfeifer. *Qualitätsmanagement – Strategien, Methoden und Techniken*. Carl Hanser Verlag, Munich, Germany, 3rd edition, 2001.

[Pos80] Jon Postel. *Request for Comments 768 – User Datagram Protocol (UDP)*. Information Science Institute – University of Southern California, Marina del Rey, USA, August 1980.

[Pow11] Scott Powell. *Ethernet Physical Layer Alternatives for Automotive Applications*. In Proceedings of 1st Ethernet&IP@Automotive Day, Munich, Germany, November 2011.

[Pro12] Profinet IO, April 2012. http://www.profibus.com/.

[Rah09] Mehrnoush Rahmani. *A Resource-Efficient IP-based Network Architecture for In-Vehicle Communication*. Dissertation at the Technical University of Munich, Munich, Germany, March 2009.

[Rec08] Jöerg Rech. *Ethernet – Technologien und Protokolle für die Computervernetzung*. Heise Zeitschriften Verlag GmbH & Co. KG, Hamburg, Germany, 2nd edition, 2008.

[Rei10] Norbert Reindl. *E/E-Fahrzeugarchitekturen der Zukunft*. In Proceedings of ELMOS Workshop 'Mobilität 2020 ff', Dortmund, Germany, September 2010.

[Req76] Request for Comments 707. *A High-Level Framework for Network-based Resource Sharing*. Information Science Institute – University of Southern California, Marina del Rey, USA, January 1976.

[Req81a] Request for Comments 791. *Internet Protocol Version 4 – Darpa Internet Program Protocol Specification (IPv4)*. Information Science Institute – University of Southern California, Marina del Rey, USA, September 1981.

[Req81b] Request for Comments 792. *Internet Control Message Protocol – Darpa Internet Program Protocol Specification (ICMP)*. Information Science Institute – University of Southern California, Marina del Rey, USA, September 1981.

[Req81c] Request for Comments 793. *Transmission Control Protocol – Darpa Internet Program Protocol Specification (TCP)*. Information Science Institute – University of Southern California, Marina del Rey, USA, September 1981.

[RFC12] RFC – Request for Comments – IETF Repository, September 2012. http://www.ietf.org/rfc.html/.

[RK06] Justin Ray and Philip Koopman. *Efficient High Hamming Distance CRCs for Embedded Networks*. In Proceedings of the International Conference Dependable Systems and Networks (DSN2006), Philadelphia, USA, June 2006.

[RMRHS07] Mehrnoush Rahmani, Bernd Müller-Rathgeber, Wolfgang Hintermaier, and Eckehard Steinbach. *Error Detection Capabilities of Automotive Network Technologies and Ethernet - A Comparative Study -*. In Proceedings of the IEEE Intelligent Vehicles Symposium (IV2007), Istanbul, Turkey, January 2007.

[RNP04] Guillermo Rodríguez-Navas and Julían Proenza. *Clock Synchronization in CAN Distributed Embedded Systems*. In Proceedings of International Workshop on Real-Time Networks (RTN2004), Catania, Italy, 2004.

[SBLS11] Thilo Streichert, Stefan Buntz, Helmut Leier, and Stefan Schmerler. *Short and Long Term Perspective for Ethernet for Vehicle-internal Communication*. In Proceedings of 1st Ethernet&IP@Automotive Day, Munich, Germany, November 2011.

[Ser12] Sercos III, April 2012. http://www.sercos.de/.

[Sin11] Stefan Singer. *Freescale Enabled Automotive Ethernet Applications*. In Proceedings of 1st Ethernet&IP@Automotive Day, Munich, Germany, November 2011.

[SKSB11] Christoph Schmutzler, Andreas Krüger, Martin Simons, and Jürgen Becker. *Ansätze zur Integration von energieeffizienten Intelligenten Kommunikationskontrollern für FlexRay in Autosar*. In Proceedings of GI Workshop Automotive Software Engineering, Berlin, Germany, October 2011.

[SM05] Frank Schiller and Tina Mattes. *An Efficient Method to Evaluate CRC-Polynomials for Safety-Critical Industrial Communication*. In Proceedings of the 11th International Conference on System Modelling and Control (SMC2005), Zakopane, Poland, 2005.

[Spi11] Spirent Communications. *Spirent – TestCenter*, February 2011. http://www.spirent.com/.

[ST12] Thilo Streichert and Matthias Traub. *Elektrik/Elektronik-Architekturen im Kraftfahrzeug*. Springer Verlag Berlin Heidelberg, Germany, 1st edition, 2012.

[Sys11] SysTec-Electronics. CAN/Ethernet-Gateway, http://www.systec-electronic.com/, 2011.

[Tel12] Telemotive AG. *Blue PiraT – Datalogger*, March 2012. http://www.telemotive.de/.

[TG08] Michael Johas Teener and Geoffrey Garner. *Overview and Timing Performance of IEEE802.1AS*. In Proceedings of the IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS2008), Ann Arbor, USA, pages 49 – 53, September 2008.

[Tim12] Time-Triggered Ethernet (TTEthernet), April 2012. http://www.tttech.com/.

[Tra10] Matthias Traub. *Durchgängige Timing-Bewertung von Vernetzungsarchitekturen und Gateway-Systemen im Kraftfahrzeug*. Karlsruher Institut für Technologie (KIT), Karlsruhe, Germany, 2010.

[Vec09] Vector Technical Article. *The Optimal Operating System for FlexRay-based Applications*. In Proceedings of Elektronik Automotive 9/2009, September 2009.

[Vec11] Vector Informatik GmbH. *Vector – CANoe.IP*, February 2011. http://www.vector.com/.

[Vec12a] Vector Informatik GmbH. DBC-Kommunikationsdatenbasis für CAN, http://www.vector.com/vi_candb_de.html, 2012.

[Vec12b] Vector Informatik GmbH. *FlexCard Cyclone II / E-Ray*, October 2012. https://www.vector.com/vi_flexcard_en.html/.

[VK08] Alexandre Van Kempten. *IEEE1588v1 vs. IEEE 1588v2*, 2008.

[Völ11] Lars Völker. *One for All – Interoperability from AUTOSAR to Genivi*. In Proceedings of 1st Ethernet&IP@Automotive Day, Munich, Germany, November 2011.

[Zim80] Hubert Zimmermann. *OSI Reference Model – The OSI Model of Architecture for Open Systems Interconnection*. In Proceedings of the IEEE Transactions on Communications, Volume 28, No. 4, pages 425 – 432, April 1980.

[ZM07] Jian Zhang and Andrew Moore. Traffic trace artifacts due to monitoring via port mirroring. In *Workshop on End-to-End Monitoring Techniques and Services (E2EMON2007)*, Munich, Germany, May 2007.

Author's Own Publications

[Ker08•] Andreas Kern. *Theoretische und experimentelle Analyse des Internet-Protokolls als Kommunikationsmedium im Automobil*. Diplomarbeit, Lehrstuhl für Informatik 12 – Hardware-Software-Co-Design, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany, December 2008.

[KRST11•] Andreas Kern, Dominik Reinhard, Thilo Streichert, and Jürgen Teich. *Gateway Strategies for Embedding of Automotive CAN-frames into Ethernet-packets and Vice Versa*. In Proceedings of the Architecture of Computing Systems (ARCS2011), Como, Italy, February 2011.

[KSG+11•] Andreas Kern, Thilo Streichert, Nandhini Ganesan, Josef Nöbauer, and Helge Zinner. *FlexRay/Ethernet Transport Protocol Concept*. IEEE 1722 Working Group Contribution, September 2011.

[KSS+10•] Andreas Kern, Christoph Schmutzler, Thilo Streichert, Michael Hübner, and Jürgen Teich. *Network Bandwidth Optimization of Ethernet-based Streaming Applications in Automotive Embedded Systems*. In Proceedings of the International Conference on Computer Communication Networks (ICCCN2010), Zurich, Switzerland, August 2010.

[KST11•] Andreas Kern, Thilo Streichert, and Jürgen Teich. *An Automated Data Structure Migration Concept – From CAN to Ethernet/IP in Automotive Embedded Systems (CANoverIP)*. In Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE2011), Grenoble, France, March 2011.

[KSZ11•] Andreas Kern, Thilo Streichert, and Hongyan Zhang. *Verfahren zum Mitlesen und Versenden von Nachrichten sowie zur Simulation eines*

Netzwerkes eines Fahrzeuges. Offenlegungsschrift DE 10 2010 054 093 A1, Deutsches Patent- und Markenamt, August 2011.

[KZS+11•] Andreas Kern, Helge Zinner, Thilo Streichert, Josef Nöbauer, and Jürgen Teich. *Accuracy of Ethernet AVB Time Synchronization Under Varying Temperature Conditions for Automotive Networks*. In Proceedings of the Design Automation Conference (DAC2011), San Diego, USA, June 2011.

[KZST11•] Andreas Kern, Hongyan Zhang, Thilo Streichert, and Jürgen Teich. *Testing Switched Ethernet Networks in Automotive Embedded Systems*. In Proceedings of the International Symposium on Industrial Embedded Systems (SIES011), Västerås, Sweden, June 2011.

[RKH+10•] Felix Reimann, Andreas Kern, Christian Haubelt, Thilo Streichert, and Jürgen Teich. *Echtzeitanalyse Ethernet-basierter E/E-Architekturen im Automobil*. In Proceedings of the Automotive meets Electronics Conference (AME2010), Dortmund, Germany, April 2010.

[SKB+10•] Thilo Streichert, Andreas Kern, Stefan Buntz, Helmut Leier, and Stefan Schmerler. *Ethernet/IP in E/E-Architectures – Motivation, Applications, and Current State*. In Proceedings of Hanser ProductDay Automotive Networks and Software Architectures, Stuttgart, Germany, November 2010.

[SKB+11•] Thilo Streichert, Andreas Kern, Stefan Buntz, Helmut Leier, and Stefan Schmerler. *Ethernet for Vehicle-internal Communication – Motivation, Application Scenarios and Current State*. At the Freescale Technology Forum – Americas (FTF2011), San Antonio, USA, June 2011.

Acronyms

ACK . . . . . Acknowledgment
ADC . . . . . Analogue Digital Converter
AFDX . . . . . Avionics Full-Duplex Switched Ethernet
API . . . . . Application Programming Interface
ARP . . . . . Address Resolution Protocol
AUI . . . . . Attachment Unit Interface
AVB . . . . . Audio Video Bridging
BCI . . . . . Bulk Current Injection
BE . . . . . Bit Error
BER . . . . . Bit Error Rate
BMCA . . . . . Best Master Clock Algorithm
CAN . . . . . Controller Area Network
CC . . . . . Communication Controller
CM . . . . . Clock Master
COTS . . . . . Commercial Off The Shelf
CPU . . . . . Central Processing Unit
CRC . . . . . Cyclic Redundancy Check
CS . . . . . Clock Slave
CSMA/CA . . . . . Carrier Sense Multiple Access Collision Avoidance
CSMA/CD . . . . . Carrier Sense Multiple Access Collision Detection
CSMA/CR . . . . . Carrier Sense Multiple Access Collision Resolution
DAC . . . . . Digital Analogue Converter
DB-DMA . . . . . Double Buffer Direct Memory Access
DDR-RAM . . . . . Double Data Rate Random Access Memory
DES . . . . . Discrete Event Simulation
DLC . . . . . Data Length Code
DLL . . . . . Dynamic Link Library

DMA ..... Direct Memory Access
DNS ..... Domain Name System
DoIP ..... Diagnostics over IP
EA ..... Evolutionary Algorithm
ECU ..... Electronic Control Unit
E/E-architecture ..... Electric and Electronic Architecture
EMC ..... Electromagnetic Compatibility
ETH ..... Ethernet
FCS ..... Frame Check Sequence
FIFO ..... First In, First Out
FPGA ..... Field Programmable Gate Array
FR ..... FlexRay
HTTP ..... Hypertext Transfer Protocol
IEEE ..... Institute of Electrical and Electronics Engineers
IFG ..... Internet Frame Gap
IP ..... Internet Protocol
IPv4 ..... Internet Protocol Version 4
IPv6 ..... Internet Protocol Version 6
IRQ ..... Interrupt
ISO ..... International Organization for Standardization
LAN ..... Local Area Network
LFSR ..... Linear Feedback Shift Register
LIN ..... Local Interconnect Network
LLL ..... Logical Link Layer
LNK ..... Link
LUT ..... Look-Up Table
LVDS ..... Low-Voltage Differential Signaling
MAC ..... Medium Access Control
MDI ..... Media Dependent Interface
MII ..... Media Independent Interface
MOST ..... Media Oriented Systems Transport
MTIE ..... Maximum Time Interval Error
MTU ..... Maximum Transmission Unit
NIC ..... Network Interface Card

OEM ..... Original Equipment Manufacturer
OS ..... Operating System
OSI ..... Open Systems Interconnection
PDU ..... Physical Data Unit
PE ..... Packet Error
PER ..... Packet Error Rate
PHY ..... Physical Layer
PLS ..... Physical Layer Signaling
PMA ..... Physical Medium Attachment
PTP ..... Precision Time Protocol
QoS ..... Quality of Service
RAM ..... Random Access Memory
RER ..... Residual Error Rate
RFC ..... Request For Comment
RPC ..... Remote Procedure Call
RPE ..... Residual Paket Error
RR ..... Round Robin
RX ..... Receive
SNMP ..... Simple Network Management Protocol
SD ..... Service Discovery
SG-DMA ..... Scatter Gather Direct Memory Access
SoC ..... System on Chip
SOP ..... Start Of Production
SRAM ..... Static Random Access Memory
STP ..... Shielded Twisted Pair
STQ ..... Shielded Twisted Quad
SWC ..... Software Component
SYNC ..... Synchronization
TCIM ..... Tightly Coupled Instruction Memory
TCP ..... Transmission Control Protocol
TDMA ..... Time Division Multiple Access
TOS ..... Type Of Service
TTL ..... Time To Life
TX ..... Transmit

UDP . . . . . User Datagram Protocol
UTP . . . . . Unshielded Twisted Pair
VLAN . . . . . Virtual Local Area Network

Symbols

α ..... cycle time factor
b_{eth} ..... Ethernet throughput
b_{ip} ..... IP throughput
b_{udp} ..... UDP throughput
b_s ..... buffer size
b_t ..... buffer timeout
$\mathbf{d}$ ..... data vector
e ..... ECU
E ..... set of ECUs
$\mathbf{e}$ ..... error vector
ECU_{rx} ..... set of receivers
ECU_{tx} ..... set of transmitters
f ..... frame
F ..... set of frames
$\mathcal{F}$ ..... function
i ..... index/position
l ..... length/size
p ..... packet
P ..... set of packets
P_{bit} ..... bit error probability
P_{re} ..... residual error probability
s ..... signal
S ..... set of signals
t ..... point at time or timestamp
t^{ct} ..... cycle time
t^{cta} ..... cycle time active
t^{dt} ..... delay time

t^o .... time offset

$\oplus$ .... XOR

Index

accuracy
    synchronization, 41
AVB, 18

bit error, 21
bit error rate, 21

concept
    error detection, 27
conclusion, 171
contributions, 5

definition
    bit error, 21
    bit error rate, 21
    error types, 32
    packet error, 21
    packet error rate, 22
    residual error, 21
    residual error probability, 22
    residual error rate, 22

error rate
    determination, 21
error types, 32
Ethernet
    AVB, 18
    history, 13
    standards, 14
Ethernet-Test-Switch, 155
Ethernet/IP
    requirements, 12

gateway, 90
    CAN/Ethernet, 91
    FlexRay/AVB, 105

IEEE
    1588-2002, 43
    1588-2008, 44
    802.1, 18
    802.1AS, 44
    1722, 18
IP, 16

measurement
    BCI, 33

optimization
    throughput, 58

packet error, 21
packet error rate, 22
protocol

IEEE 1722, 18
TCP, 18
UDP, 17

replacement, 121
CAN, 121
FlexRay, 133
residual error, 21
residual error probability, 22
residual error rate, 22

simulation, 147
restbus, 155
synchronization
physical-shared, 41
switched-networks, 43

TCP, 18

UDP, 17

Curriculum Vitae

In 2003, Andreas Kern started his studies in Computer Science at the Friedrich Alexander University of Erlangen-Nuremberg, Germany. After doing his diploma thesis at Daimler AG in Sindelfingen, Germany, he received his diploma in 2008. During his studies, he worked as a student assistant and tutor in the department of Hardware-Software-Co-Design. From 2009 to 2011, Andreas Kern was a researcher and industrial PhD candidate at the Daimler AG in Sindelfingen, Germany, in cooperation with the department of Computer Science at the University of Erlangen-Nuremberg. Since the end of 2011, Andreas Kern is working at the centre of research and innovations of BMW in Munich, Germany.