A GENERAL FRAMEWORK FOR THE DETERMINISTIC MEDIUM ACCESS ON THE CONTROLLER AREA NETWORK

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MURAT AKPINAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JUNE 2022

Approval of the thesis:

A GENERAL FRAMEWORK FOR THE DETERMINISTIC MEDIUM ACCESS ON THE CONTROLLER AREA NETWORK

submitted by **MURAT AKPINAR** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** ________________

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** ________________

Prof. Dr. Ece Güran Schmidt
Supervisor, **Electrical and Electronics Engineering, METU** ________________

Prof. Dr. Klaus Werner Schmidt
Co-supervisor, **Electrical and Electronics Engineering, METU** ________________

Examining Committee Members:

Prof. Dr. Halit Oğuztüzün
Computer Engineering, METU ________________

Prof. Dr. Ece Güran Schmidt
Electrical and Electronics Engineering, METU ________________

Prof. Dr. Bülent Tavlı
Electrical and Electronics Engineering, TOBB ETÜ ________________

Prof. Dr. Ali Ziya Alkar
Electrical and Electronics Engineering, Hacettepe University ________________

Assoc. Prof. Dr. Mustafa Mert Ankaralı
Electrical and Electronics Engineering, METU ________________

Date: 13.06.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Murat Akpınar

Signature :

ABSTRACT

A GENERAL FRAMEWORK FOR THE DETERMINISTIC MEDIUM ACCESS ON THE CONTROLLER AREA NETWORK

Akpınar, Murat
Ph.D., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Ece Güran Schmidt
Co-Supervisor: Prof. Dr. Klaus Werner Schmidt

June 2022, 192 pages

This thesis proposes a general framework CANDS (Controller Area Network with Determinism and Synchronization support) for in-vehicle communication on CAN. CANDS features a hierarchy of novel clock synchronization (CS) algorithms with different levels of clock accuracy and implementation complexity. Moreover, CANDS realizes our original idea of weak time division multiple access (WTDMA) for deterministic bus access, while being fully compatible with the standard CAN protocol and its recent extensions.

CS methods within CANDS apply offset and drift correction. While offset correction is performed based on timestamps in periodic reference messages (RMs), drift correction is realized based on (1) the re-synchronization mechanism of the CAN bit timing; (2) drift estimates computed from periodic timestamps with RMs and (3) drift estimates determined in a discrete-time feedback control loop. Since accurate timestamps are essential for the CS performance, the thesis further develops a new timestamping (TS) method, which reduces the effect of uncertainties that are caused by the CAN bit timing, oscillator drifts and different cable lengths. Overall, clock accuracies below 100 ns are achieved in hardware experiments with our TS method and CS algorithms.

Benefiting from the achieved clock accuracy levels, the thesis next introduces WTDMA as a new method for slotted medium access on CAN together with novel message scheduling algorithms for the assignment of time slots in WTDMA. WTDMA is realized in software and a certain degree of interference between adjacent time slots is tolerated with the usage of Carrier Sense Multiple Access/Collision Resolution (CSMA/CR) and the non-preemptive message transmission on CAN. Sufficient conditions for the correct operation of WTDMA are derived and its practicability is validated with comprehensive hardware experiments. Specifically, bus loads above 90% and deterministic message latencies in the order of hundreds of microseconds are achieved.

Keywords: controller area network, clock synchronization, drift compensation, experimental evaluation, timestamping, slotted bus access, determinism.

ÖZ

CAN AĞI ÜZERİNDE DETERMİNİSTİK ORTAM ERİŞİMİ İÇİN GENEL BİR İŞ ÇERÇEVESİ

Akpınar, Murat
Doktora, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Ece Güran Schmidt
Ortak Tez Yöneticisi: Prof. Dr. Klaus Werner Schmidt

Haziran 2022 , 192 sayfa

Bu tez ile genel bir iş çerçevesi olarak CANDS (Determinizm ve Senkronizasyon destekli Kontrolör Alan Ağı), CAN üzerinde araç içi iletişim için önerilmektedir. CANDS, farklı saat doğruluğu ve uygulama karmaşıklığına sahip yeni saat senkronizasyonu (CS) algoritmalarından oluşan bir hiyerarşiye sahiptir. CANDS, standart CAN protokolü ve son uzantıları ile tamamen uyumluyken, deterministik veri yolu erişimi için zayıf zaman bölmeli çoklu erişim (WTDMA) konusundaki orijinal fikrimizi gerçekleştirir.

CANDS içerisindeki CS yöntemleri, ofset ve kayma düzeltmesi uygular. Ofset düzeltmesi periyodik referans mesajları (RM) ile taşınan zaman damgalarına göre yapılırken, kayma düzeltmesi (1) CAN bit zamanlamasının yeniden senkronizasyon mekanizmasına; (2) periyodik zaman damgalarından hesaplanan kayma tahminlerine ve (3) ayrık zamanlı bir geri besleme kontrol döngüsünde belirlenen kayma tahminlerine göre gerçekleştirilir. CS performansı için doğru zaman damgaları gerekli olduğundan, bu tez ayrıca CAN bit zamanlamasının, osilatör kaymalarının ve farklı kablo uzunluklarının neden olduğu belirsizliklerin etkisini azaltan yeni bir zaman damgası (TS) yöntemi de geliştirir. Genel olarak, TS yöntemimiz ve CS algoritmalarımız ile yapılan donanım deneylerinde 100 ns ve altında saat doğrulukları elde edilmektedir.

Elde edilen saat doğruluğu sayesinde, CAN üzerinde ayrılmış ortam erişimi için WTDMA ve WTDMA zaman dilimlerinin atanması için yeni mesaj çizelgeleme algoritmaları önerilmektedir. WTDMA yazılımda gerçekleştirilir ve bitişik zaman dilimleri arasında belirli bir düzeyde girişime Taşıyıcı Algılama Çoklu Erişim/Çarpışma Çözümleme (CSMA/CR) ve CAN önleyici olmayan mesaj iletimi sayesinde izin verilir. WTDMA fikrinin doğru çalışması için yeterli koşullar ortaya konulmuş ve kapsamlı donanım deneyleri ile uygulanabilirliği doğrulanmıştır. Spesifik olarak, %90 üzerinde veri yolu yükleri ve birkaç yüz mikro saniye düzeyinde deterministik mesaj gecikmeleri elde edilmektedir.

Anahtar Kelimeler: controller area network, saat senkronizasyonu, kayma düzeltme, deneysel değerlendirme, zaman damgası, ayrılmış ortam erişimi, determinizm.

To my family

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisors Prof. Dr. Şenan Ece Schmidt and Prof. Dr. Klaus Werner Schmidt who have supported me continuously with their motivation and immense knowledge.

Additional to my advisors, I would like to thank Prof. Dr. Halit Oğuztüzün, Assoc. Prof. Dr. Mustafa Mert Ankaralı and Prof. Dr. Cüneyt F. Bazlamaçcı for their positive attitudes and supportive comments during my thesis monitoring meetings. Moreover, I also thank the jury members of the thesis defense, Prof. Dr. Ali Ziya Alkar and Prof. Dr. Bülent Tavlı for their insightful comments and questions.

I would also like to express my gratitude to ASELSAN for the financial support to attend conferences and to take graduate courses. I also thank TUBITAK since some of the studies conducted in this thesis are supported by the Scientific and Technological Research Council of Turkey (TUBITAK) [Project Code 119E277].

I need to thank my friends, Ali, Savaş, Muhammet, Ahmet, Emre, Soysal, Tuğrul and Çağrı for their friendship for several years. In addition, I am also thankful to my friends at ASELSAN Sami, Bora, Çağrı, Onur, Koray and Ayhan for their enjoyable friendship.

I want to thank my second family -Sönmez family- for their encouraging support and warm behaviours. I should also give special credit to my mother Nezahat Akpınar, my father Ahmet Akpınar and my brother Mehmet Akpınar for their invaluable support in my entire life.

Above all, I would like to present my gratitude to my wife Gizem Sönmez Akpınar for her peerless support and I am grateful to every moment she stands by me. In my last words, I would like to express my love to our baby girl Eylül who will join our family soon.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xvii

LIST OF FIGURES xix

LIST OF ABBREVIATIONS xxiii

CHAPTERS

1 INTRODUCTION 1

2 PRELIMINARY WORK 7

  2.1 CAN Protocol 7

    2.1.1 The Basic Concepts of CAN 8

    2.1.2 CAN Message Transfer 9

    2.1.3 CAN Bit Timing 11

    2.1.4 CAN FD Protocol 13

    2.1.5 CAN XL Protocol 14

  2.2 Clock Synchronization on CAN and Related Work 15

    2.2.1 Local Clock and Clock Drift 15

2.2.2 Clock Synchronization on CAN 16

2.2.3 Gergeleit's Method 19

2.2.4 AUTOSAR Method 21

2.3 Experimental Setup 22

2.3.1 Evaluation of the fundamental CS methods 23

2.4 Timestamping on CAN and Related Work 25

2.5 Notation 26

3 THE GENERAL FRAMEWORK 29

3.1 Message Scheduling on CAN 30

3.1.1 Classical Scheduling and WCRT computation 30

3.1.2 Offset Scheduling 31

3.2 TDMA and TTCAN 32

3.2.1 TDMA 32

3.2.2 TTCAN 33

3.3 Discussion 34

3.4 CANDS Framework 34

3.4.1 Overview 35

3.4.2 Parameter Discussion 36

4 TIMESTMAPING ON CAN 39

4.1 Background 40

4.2 Components of Timestamping on CAN 41

4.3 Uncertainties Related to the CAN Bit Timing 42

4.4 TR Quality using the SoF bit as TR Instant 43

4.5 TR Quality using the EoM as Trigger Instant 45

4.6 Predictable TR Instant 46

4.7 Timestamping Service Quality and Discussion 48

4.8 Experimental Evaluation of Timestamping on CAN 50

4.8.1 Experimental Setup 50

4.8.2 Evaluation of the Timestamping Service Quality 52

4.8.3 The Evaluation of the SoF Bit as TR Instant 54

4.8.4 The Evaluation of the EoM as TR Instant 55

4.8.5 Evaluation of the Predictable TR Instant 56

4.8.6 Overall Comparison 57

4.9 Discussion and Conclusion 57

5 SOFTWARE-BASED CLOCK SYNCHRONIZATION METHODS ON CAN 61

5.1 Revisiting the AUTOSAR Offset Correction 62

5.2 Revisiting the Gergeleit Offset Correction 64

5.3 Improved Software-based Clock Synchronization Method on CAN . . 65

5.3.1 Improved Software-based Clock Synchronization 66

5.3.2 Analysis of the ISCS Method 70

5.3.3 Evaluation of the ISCS Method 71

5.3.3.1 Experimental Setup 71

5.3.3.2 Experimental Results 72

5.4 RM-based Drift Correction Clock Synchronization for the Controller Area Network 75

5.4.1 Drift Estimation based on RMs 75

5.4.2 Experimental Setup and Evaluation 77

5.5 Controller-based Clock Synchronization for the Controller Area Network 78

5.5.1 Basic Idea 78

5.5.2 Discrete-time Control Problem 80

5.5.3 Experimental Setup 81

5.5.4 Basic Evaluation of the Existing CS methods 82

5.5.5 Implementation Details and Evaluation 83

5.5.6 Initialization Phase 85

5.5.7 Further Evaluation with Initialization Phase 86

5.5.8 AUTOSAR CS Compatible Controller Model and Parameters . 88

5.6 Dependency of the CS on Timestamping Quality 89

5.6.1 Experimental Evaluation 90

5.7 Discussion and Conclusion 93

6 CLOCK SYNCHRONIZATION FOR THE CONTROLLER AREA NETWORK USING BIT TIMING INFORMATION 95

6.1 Basic Idea behind the Phase-error Based Drift Correction 96

6.2 Usage of the Approximate GCD 98

6.3 Approximate GCD-based Drift Correction 100

6.4 Experimental Evaluation 104

6.4.1 Implementation Details 104

6.4.2 Comprehensive Experimental Evaluation 106

6.4.3 Start-up Phase 110

6.5 Start-up Method for ACS-PEDC 111

6.5.1 Start-up Method 112

6.5.2 Start-up Evaluation 114

6.6 Discussion and Conclusion 115

7 WEAK TDMA ON CAN 117

7.1 Assumptions 118

7.2 Protocol Definition 119

7.3 Sufficient Conditions for Correct Operation of WTDMA 120

7.4 Advantages of Our Method 125

7.5 Experimental Evaluation 126

7.5.1 Experimental Setup and Implementation of WTDMA 126

7.5.2 Parameters and Performance Metrics 127

7.5.3 Message Set 127

7.5.4 TDMA Operation and WCRTs 128

7.5.5 Evaluation of Computed Bounds 130

7.5.6 Usage of a Reduced Window Size 132

7.5.7 Independence of the CS Method 134

7.5.8 CAN Bus with Fault Injection 134

7.5.9 CAN Nodes with Non-Sync Messages 135

7.6 Discussion and Conclusion 137

8 WTDMA MESSAGE SCHEDULING 139

8.1 Background 140

8.2 Integer Linear Programming for WTDMA Scheduling 141

8.2.1 ILP for the First Step - Mini Time Slot Assignment 141

8.2.2 ILP for the Second Step - Ultimate Offset Assignment 144

8.3 The Heuristic Method for WTDMA Scheduling 149

8.3.1 The Heuristic Method for the First Step - Mini Time Slot Assignment 149

8.3.2 The Heuristic Method for the Second Step - Ultimate Offset Assignment 151

8.4 Evaluation of the WTDMA Scheduling Methods 153

8.5 Discussion 156

9 CONCLUSION 159

REFERENCES 163

A PROOFS IN CHAPTER 5 177

A.1 Clock Accuracy of Offset Correction 177

B PROOFS IN CHAPTER 6 181

B.1 Proof of Theorem 2 181

B.2 Proof of Theorem 3 183

B.3 Clock Accuracy of ACS-PEDC 185

B.4 Start-up algorithm 187

C PROOFS IN CHAPTER 7 189

C.1 Proof of Theorem 5 189

CURRICULUM VITAE 191

LIST OF TABLES

TABLES

Table 2.1 Bounds on the frame bit length depending on B_i. 27

Table 4.1 Dependency comparison. 49

Table 4.2 The trigger quality comparison. 59

Table 5.1 Clock accuracy results. 91

Table 5.2 Clock accuracy - overall comparison. 92

Table 6.1 Comparison of Δc_{S3} for different experiments. 107

Table 6.2 Maximum and mean PEO period detection times. 111

Table 6.3 Evaluation of $t_{2\,\mu\mathrm{s}}$ depending on U_{TM} and d_{S3}. 115

Table 7.1 Message and schedule properties. 129

Table 7.2 Observed ratios of schedule violations. 131

Table 7.3 Properties of the modified messages. 133

Table 8.1 Example of CAN message set. 143

Table 8.2 The s_i values found by ILP. 144

Table 8.3 Example of CAN message set. 147

Table 8.4 Comparison of the success rate of the first step methods. 154

Table 8.5 Comparison of the run-times of the first step methods. 155

Table 8.6 The run-time comparison for H-S2 and ILP-S2. 156

LIST OF FIGURES

FIGURES

Figure 2.1 CAN frame - standard format. 10

Figure 2.2 CAN frame - extended format. 10

Figure 2.3 CAN bit segments. 12

Figure 2.4 CAN-FD frame - standard format (11 bit ID). 13

Figure 2.5 CAN-FD frame - extended format (29 bit ID). 14

Figure 2.6 Illustration of RM transmissions compatible to Gergeleit's method. 19

Figure 2.7 Illustration of clock difference when Gergeleit's method is followed. 20

Figure 2.8 Illustration of the RMs (SYNC and FUP) according to AUTOSAR SW-CS method. 21

Figure 2.9 The clock difference example in case of AUTOSAR CS method. 22

Figure 2.10 The general view of the experimental setup. 23

Figure 2.11 Clock difference measurement of S3: Gergeleit's method. 24

Figure 2.12 Clock difference measurement of S3: AUTOSAR. 24

Figure 3.1 TDMA with guard times. 32

Figure 3.2 Overview of the CANDS framework. 36

Figure 4.1 Signal propagation delays on the CAN bus. 40

Figure 4.2 The components of the timestamping. 41

Figure 4.3 CAN bit timing synchronization uncertainty. 42

Figure 4.4 Possible time difference due to the oscillator drift. 43

Figure 4.5 Timing analysis of SoF bit TR instant. 44

Figure 4.6 Timing analysis of EOM TR instant. 46

Figure 4.7 Timing analysis of the predictable TR instant. 48

Figure 4.8 Experimental setup. 51

Figure 4.9 Software timestamp service delays on different nodes. 53

Figure 4.10 The quality measurements. 53

Figure 4.11 SoF bit - the trigger quality evaluation. 54

Figure 4.12 EoM - the trigger quality evaluation. 55

Figure 4.13 Predictable TR method - the trigger quality evaluation. 56

Figure 5.1 Illustration of the Gergeleit's method with a timestamping inaccuracy e_{TS}. 65

Figure 5.2 The simulation with random TS errors. 65

Figure 5.3 Illustration of ISCS method. 69

Figure 5.4 Clock drift. 72

Figure 5.5 The dependency of the clock accuracy of S3 on the RM period. . 73

Figure 5.6 The dependency of the clock accuracy of S3 on the bus load. . . 74

Figure 5.7 The dependency of the clock accuracy of S3 on the bit rate. . . . 74

Figure 5.8 The clock accuracy of S3 for different CS methods. 75

Figure 5.9 Computation of the LC update interval. 76

Figure 5.10 Clock drift measurement for the experimental setup. 77

Figure 5.11 Clock difference for the AUTOSAR method. 78

Figure 5.12 AUTOSAR-DC: Dependency on the RM period. 79

Figure 5.13 Feedback loop. 80

Figure 5.14 Clock drift. 82

Figure 5.15 Comparison of existing methods. 83

Figure 5.16 Comparison of simulation and measurements for different values of T for $B = 250$ kbps and $U = 70\%$. 84

Figure 5.17 Comparison of the control algorithm with and without initialization for $B = 250$ kbps and $U = 70\%$. 86

Figure 5.18 Clock differences for different parameters. 87

Figure 5.19 The clock difference results. 90

Figure 5.20 The clock difference results. 93

Figure 6.1 Illustration of the PEO detection. 98

Figure 6.2 Example iterations of Algorithm 7 with table v. 101

Figure 6.3 The components of ACS-PEDC. 105

Figure 6.4 Measurements for different RM periods. 108

Figure 6.5 Measurements for a time-varying clock drift. 110

Figure 6.6 Start-up measurements for different drift values. 115

Figure 7.1 WTDMA notation and example illustration. 122

Figure 7.2 Minimum window size and minimum frame length. 124

Figure 7.3 Starting time delays for a bit rate of $B = 250$ kbps. 130

Figure 7.4 Starting time delays for a bit rate of $B = 1$ Mbps. 131

Figure 7.5 The distribution of starting times of messages. 132

Figure 7.6 The starting time delays of CAN messages - 250 kbps with smaller window sizes. 134

Figure 7.7 Response times of messages. 136

Figure 8.1 Example of mini time slot assignment. 145

Figure 8.2 Example of mini time slot schedule. 147

LIST OF ABBREVIATIONS

AGCD	Approximate Greatest Common Divisor
ACS-PEDC	Accurate Clock Synchronization method with Phase-Error based Drift Correction
ACK	Acknowledgement
BER	Bit Error Rate
CAN	Controller Area Network
CAN FD	CAN with Flexible Data rate
CAN XL	CAN with Extended Length
CRC	Cyclic Redundancy Check
CO	Clock Oscillator
CS	Clock Synchronization
CCIP	CAN Controller IP
COTS	Commercial off-the-shelf
CC	CAN Controller
CiA	CAN in Automation
CANDS	Controller Area Network with Determinism and Synchronization support
DLC	Data Length Code
EI	Environmental Instability
EoM	End of Message
FPGA	Field Programmable Gate Array
FUP	Follow-up
GCD	Greatest Common Divisor
HR	Hardware Register

HP	Hyper-Period
ID	Identifier
IDE	Identifier Extension
IDS	Intrusion Detection Systems
IA	Initial Accuracy
ILP	Integer Linear Programming
LC	Local Clock
LTI	Long Term Instability
NTU	Network Time Unit
NTP	Network Time Protocol
ODU	Oscillator Drift Uncertainty
PEO	Phase Error Overflow
PEOD	Phase Error Overflow Detection
PLL	Phase Locked Loop
PTP	Precision Time Protocol
ppm	parts per million
QoS	Quality of Service
RTR	Remote Transmission Request
RM	Reference Message
RJW	Re-synchronization Jump Width
RT	Response Time
RC	Reference Clock
SDCC	Software-defined CAN Controller
SoF	Start of Frame
SRR	Substitute Remote Request
SW-CS	Software-based Clock Synchronization
SIC	Signal Improvement Capability

SP	Sample Point
TS	Timestamping
TR	Trigger
TDMA	Time Division Multiple Access
TTCAN	Time Triggered CAN
TQ	Time Quantum
TQC	TQ counter
TM	Time Master
TSU	Timestamping Unit
WCRT	Worst Case Response Time
WTDMA	Weak TDMA

CHAPTER 1

INTRODUCTION

Modern vehicles comprise of several distributed electronic control units (ECUs) which require to exchange information continuously [1, 2, 3]. The communication between ECUs has been realized through the bus-based in-vehicle networks such as controller area network (CAN) [4], FlexRay [5], local interconnect network (LIN) [6] and media oriented systems transport (MOST) [7]. Furthermore, the complexity and diversity of the electronic devices continue to increase together with advanced safety, driver assistance and infotainment functions in new vehicles [1]. Thus, the deployment of various new ECUs has accelerated the demand for more efficient and high bandwidth in-vehicle communication solutions [3, 8, 9].

CAN is the most widespread in-vehicle network in modern cars. It supports a maximum bus speed of 1 Mbps at lengths of up to 40 m. Moreover, CAN provides high reliability at a low cost and low power consumption [1, 10, 11]. Thus, CAN is indispensable especially for the transmission of the data of real-time control applications which require low bandwidth but high real time Quality of Service (QoS) such as ABS, suspension and braking systems [1]. Furthermore, several leading automotive producers such as Volkswagen, BMW, Porsche, Audi and Tesla still continue to use CAN in their cars today [12]. Specifically, CAN is used in the drive system and adaptive cruise control system of Audi A-8 D5 which is the world's first level 3 autonomous car [12]. In addition, Tesla Model S/X/3 which are another leading autonomous cars also adopt CAN in their powertrain, chassis and body control systems [12].

The latest researches regarding the in-vehicle networks also show interest in the wireless solutions together with the automotive ethernet [11, 12]. The wireless communication between ECUs is discussed in the literature [2, 13] as an alternative to the wired in-vehicle networks. Even though it may offer an advantage in terms of cabling usage, the wireless in-vehicle communication does not seem feasible in the near future due to the concerns about its security and dependability for safety critical automotive systems [1]. On the other hand, it seems that Ethernet is very likely to be a significant component of the future in-vehicle networks even though it is not very widespread for now [1, 14, 11, 3]. Since Ethernet does not provide deterministic message transmission, the approaches IEEE 802.1Q, Auido Video Bridging (AVB) and TTEthernet [15] have been proposed to improve the real time communication capabilities of Ethernet for automotive networks. Nevertheless, ongoing studies show that Ethernet will not replace the existing communication links such as CAN and FlexRay, but rather performs together with them as the backbone for the next generation of in-vehicle communication links [1, 3].

Due to the continuing importance of CAN, the improvements on it are still ongoing. In order to increase the capabilities of the legacy CAN protocol, CAN with flexible data-rate (CAN FD) is proposed by Bosch [16] that enables higher bit rates up to 8 Mbps and longer message payloads as big as 64 bytes. Furthermore, CAN extended length (CAN XL) is still being studied to reach data rates more than 10 Mbps with message sizes up to 2048 bytes. Although the demand of higher bus speeds and the support for longer message sizes are addressed with the legacy CAN extensions CAN XL and CAN FD, the lack of clock synchronization (CS) between distributed nodes and security precautions are not considered in any of the CAN protocols yet. Therefore, there are several recent studies in the literature which introduce CS methods on CAN [17, 18, 19, 20] and intrusion detection systems (IDS) [21, 22, 23, 24, 25] for security of CAN. Furthermore, the support for deterministic bus access is introduced with Time Triggered CAN (TTCAN) protocol [26] on top of CAN. However, TTCAN is not used in the production cars, to the best of our knowledge, since it requires specific TTCAN controllers which are not compatible to the already deployed CAN networks.

In this thesis, Controller Area Network with Determinism and Synchronization support (CANDS) framework is proposed. CANDS is fully compatible with the legacy CAN protocol and can be realized with the existing standard CAN controllers. It consists of different layers mainly to provide CS and deterministic bus access. Moreover, it can perform with different timestamping and CS methods. That is, different realizations of CANDS are possible according to the available hardware resources.

Within CANDS, there is a clock service unit that realizes the local time for the node and also offers a timestamping unit. CS methods as well as different timing-based intrusion detection systems on CAN generally depend on timestamps which are expected to be taken simultaneously at different nodes. In this thesis, a detailed analysis of existing TS methods which use the start-of-frame (SoF) bit and the end-of-message (EoM) reveals several shortcomings of these methods. Accordingly, a new predictable (TS) method is proposed by exploring and mitigating the identified disadvantages. Furthermore, hardware experiments are conducted to validate our analysis results. As the main outcome, our new TS method provides better TS quality than the existing TS methods.

Regarding CS, this thesis introduces several new methods which enable advancements in terms of both offset correction and drift correction. Specifically, CS methods for CAN generally are based on periodic RMs that broadcasts the time information among the CAN nodes. The slave nodes perform offset correction whenever they receive the correct time information with RMs. In addition, the slave nodes may also estimate the clock drift with different approaches and hence perform drift correction in order to prevent the inevitable clock drift between RMs. Regarding the offset correction, AUTOSAR CS [27] and Gergeleit's method [28] are two leading CS methods on CAN. In this thesis, an Improved Software-based Clock Synchronization (ISCS) method is proposed that presents advantages by benefiting from the ongoing message transmission on CAN bus different from the existing methods. That is, ISCS method provides clock accuracy as well as AUTOSAR CS method while consuming the half of the bandwidth of AUTOSAR CS method. Furthermore, ISCS enables much better clock accuracy than Gergeleit's method even though it has the same bandwidth usage with Gergeleit's method. Additionally, clock drift estimation and drift correction methods are also developed in this thesis in order to mitigate the drift between RMs. Firstly, the drift estimation which is simply based on timestamps with RMs (RM-based CS) is introduced. Secondly, the controller-based CS methods which estimate the drift with a discrete-time feedback control loop are also developed. Thirdly, a novel Accurate Clock Synchronization with Phase Error based Drift Correction (ACS-PEDC) CS method that benefits from the internal bit timing of CAN is presented. The novelty of ACS-PEDC comes from the usage of inherent bit synchronization mechanisms of CAN protocol to estimate the clock drift, different than the other methods that are based on periodic timestamps. Thus, ACS-PEDC is not vulnerable to the TS inaccuracy while estimating the clock drift that is one of the important contributions. Furthermore, it has to be noted that the comprehensive hardware experiments are realized to present and compare the achieved clock accuracy results for the existing and proposed CS methods in this thesis. In addition, the dependency of CS performance on TS quality is experimentally shown for the first time in the literature.

Additional to the advanced CS methods, a novel weak time division multiple access (WTDMA) model is proposed in this thesis in order to realize deterministic medium access on CAN. In particular, the deterministic medium access on CAN is desirable since it enables efficient bandwidth utilization and deterministic message RTs. Our WTDMA does not need to apply the guard times which are proposed for the conventional time division multiple access (TDMA) to handle the clock drift since WTDMA depends on the specific CAN protocol features such as the usage of Carrier Sense Multiple Access/Collision Resolution (CSMA/CR) and the non-preemptive message transmission on CAN. That is, the certain degree of conflicts between the successive time slots are allowed in our WTDMA and thus it is called as *Weak* TDMA. Furthermore, WTDMA model also considers the software delays additional to the clock inaccuracy which can result in that a message may miss the starting time of its allocated window in a WTDMA schedule. Thus, our WTDMA model can be realized in software and can work with existing standard CAN controllers, different from TTCAN. Specifically, the sufficient conditions for successful WTDMA implementation is provided such that the achieved clock accuracy plus the maximum software delay should be less than the minimum message window size. When the given condition is satisfied, the order of the messages according to the offline computed WTDMA schedule is guaranteed to be followed on the bus and hence the RTs of the messages is mostly determined by the transmission times of the messages which is desired with the deterministic bus access. That is, the message RTs in the order of hundreds of microseconds are achievable for the practical bit rates of 250 kbps. Moreover, WTDMA can cooperate with any existing CS methods and the requirement for CS accuracy loosens for lower CAN bit rates since the minimum message window size gets longer. In summary, WTDMA model is the first time triggered enhancement of CAN protocol that presents a flexible TDMA model which depends on the achievable clock accuracy, the maximum software delays and minimum message window sizes. Moreover, it has to be noted that the practicability of the WTDMA is confirmed with detailed hardware experiments where bus loads up to 100% is achieved successfully with deterministic bus access on CAN.

After the introduction of WTDMA model, WTDMA scheduling methods which are used offline to determine the allocated windows for each message are required. Thus, WTDMA scheduling methods that define the windows and assign CAN messages to those windows are also developed in this thesis. Specifically, the WTDMA scheduling problem is first formulated as Integer Linear Programming (ILP) problem. Secondly, the heuristic methods are developed as alternative to ILP-based WTDMA scheduling method. Moreover, it is shown that the proposed WTDMA scheduling methods are able to provide feasible results with several challenging CAN message set examples which are constructed in line with the practical CAN applications.

It further has to be noted that this thesis covers materials that is either published in a paper or submitted for publication. The main contributions of the thesis are listed together with the relevant chapters and publications as follows:

- Chapter 4: M. Akpınar and K. W. Schmidt, "Predictable Timestamping for the Controller Area Network Evaluation and Effect on Clock Synchronization Accuracy," IEEE Transactions on Systems, Man and Cybernetics: Systems, Under Review.

- Chapter 5: M. Akpınar, E. G. Schmidt, and K. W. Schmidt, “Evaluation of clock synchronization algorithms for controller area network,” in Signal Processing and Communications Applications Conference, pp. 1–4, May 2020 [29].

- Chapter 5: M. Akpınar, K. W. Schmidt, and E. G. Schmidt, “Improved clock synchronization algorithms for the controller area network (CAN),” in International Conference on Computer Communication and Networks, pp. 1–8, IEEE, 2019 [17].

- Chapter 5: M. Akpınar, E. G. Schmidt, and K. Werner Schmidt, “Drift correction for the software-based clock synchronization on controller area network,” in 2020 IEEE Symposium on Computers and Communications (ISCC), pp. 1–6, 2020 [18].
- Chapter 6: D. E. Arkadaş, M. Akpınar, E. G. Schmidt, and K. Werner Schmidt, “Clock Synchronization for the Controller Area Network using Bit Timing Information,” in Signal Processing and Communications Applications Conference, pp. 1–4, May 2022 [30].
- Chapter 6: M. Akpınar, K. W. Schmidt, and E. G. Schmidt, "Highly Accurate Clock Synchronization with Drift Correction for the Controller Area Network," IEEE Transactions on Parallel and Distributed Systems, Accepted [31].
- Chapter 7: M. Akpınar, K. W. Schmidt, and E. G. Schmidt, "Weak TDMA for the Deterministic Medium Access on the Controller Area Network," IEEE Transactions on Intelligent Transportation Systems, Under Review.

In brief, the thesis is organized as follows. Chapter 2 introduces the background related to the CAN protocol and the existing CS and timestamping methods on CAN. The conventional message scheduling on CAN and deterministic bus access with TTCAN are explained together with our CANDS framework in Chapter 3. Chapter 4 evaluates the existing TS methods together with our new predictable TS method. In Chapter 5, ISCS, RM-based CS and the controller based CS methods are introduced. The dependency of the CS methods on TS quality is also presented in Chapter 5. Afterwards, Chapter 6 presents ACS-PEDC method that enables precise drift estimates. The WTDMA model and its verification with hardware experiments are presented in Chapter 7 and WTDMA scheduling methods are introduced in Chapter 8. Lastly, Chapter 9 presents the concluding remarks.

CHAPTER 2

PRELIMINARY WORK

In this chapter, the background related to the CAN protocol is introduced. That is, the basic principles of CAN protocol, the message formats and CAN bit timing are explained. Moreover, CAN FD and CAN XL which are extensions of the CAN protocol to provide higher bus speed and larger message payloads are also mentioned. Afterwards, the existing CS methods for CAN are introduced, since a global clock is missing in the CAN protocol. Additionally, local clock (LC) and clock drift concepts that are related to CS applications are described. The fundamental CS methods for CAN that are Gergeleit's method [28] and AUTOSAR CS [27] are explained in detail. Then, the general information about the experimental setup that is used throughout this thesis is provided and the preliminary evaluations of Gergeleit's method [28] and AUTOSAR CS method [27] are presented. The proposed CS methods on CAN depend on the usage of timestamps that are assumed to be taken simultaneously at different nodes. Thus, the existing timestamping methods for CAN and their availability in the existing CAN controllers are also discussed in this chapter. Lastly, the scheduling model notation that is followed in this thesis is presented.

2.1 CAN Protocol

CAN [4] is the most prominent in-vehicle communication bus in modern cars [32, 33, 22, 24]. Apart from the automotive industry, it has also been used in different industrial applications to provide communication between distributed components [33]. Specifically, CAN is still indispensable for safety critical components such as engine control, transmission control, braking, steering and suspension control that require high real-time quality of service (QoS) [22, 20, 1]. After the first introduction in the mid 1980s, the advancement of the legacy CAN protocol is yet ongoing with extensions such as CAN FD [16] in 2012 and CAN XL [34] starting from 2018. In particular, CAN XL will offer data rates above 10 Mbps and payloads of up to 2048 bytes [20, 34] to meet the stringent demand of contemporary real-time applications. Due to the continuing importance of CAN, the recent literature is highly interested in possible improvements for CAN such as advanced CS [17, 18, 20] and security methods [22, 32, 24]. Hereby, the recent progress in CS potentially paves the way for the deterministic medium access on CAN which is highly desired to provide efficient bandwidth utilization and deterministic message RTs [35, 36, 37].

2.1.1 The Basic Concepts of CAN

The communication on the CAN bus is realized with fixed format messages. The length of a message can change according to its payload that is allowed to be between 0 and 8 bytes. The bit rates up to 1 Mbps is supported on CAN protocol. Even though the bit rate can be set as different values in different systems according to requirements, it must be uniform and fixed within a given system. That is, the nodes are able to differentiate the bits on the bus since they know the predefined bit duration.

According to CAN protocol, any node can start sending a message when the bus is idle and the ongoing message is received by all nodes simultaneously. Moreover, each message has a unique identifier (ID) that defines the meaning of the message. That is, the nodes can decide to use or ignore the received messages according to their IDs. Additionally, all receivers that successfully receive the transmitted message notify the transmitter node by an acknowledgement.

The bus level can be logic '1' (recessive) and logic '0' (dominant). In case of simultaneous transmission by multiple nodes, the bus level is determined like a wired-AND implementation such that the bus value will be dominant if at least one node transmits a '0' bit (dominant). Additional to the wired-AND mechanism, the message priorities which are determined by unique message IDs are employed on CAN. When multiple nodes start sending messages on the bus, the bus arbitration is resolved by bitwise arbitration based on the message priorities. During the arbitration, a transmitter node reads the bus level and compares it to the bit level which it is sending. If the levels are equal, it continues with sending the next bit. If the levels are not the same, it decides that the arbitration is lost and stops transmitting the following bits. Thus, the transmitter node which is trying to send the message with the highest priority (with the lowest ID value in binary) wins arbitration and continues to send it, whereas the other nodes in the arbitration give up transmitting and start receiving the message. That is, the bus access conflict is resolved without a performance penalty regarding the data and bandwidth loss in CAN protocol.

2.1.2 CAN Message Transfer

There are four different frame types such as data frame, remote frame, error frame and overload frame. The error frame can be transmitted by a node if it detects an error on the message transfer. The overload frame can be used to provide an extra time delay between the last and the following data and remote frames. Moreover, a node can request the transmission of the data frame with the same ID by sending the remote frame. The data frame is used to transmit the data from a transmitter to the receivers. Hereby, the data and remote frames support both standard frame format and extended frame format which have 11 bit ID and 29 bit ID, respectively.

A data frame consists of seven different fields such as start of frame field, arbitration field, control field, cyclic redundancy check (CRC) field, acknowledgement (ACK) field and end of frame field. While the bus is idle, any CAN node can start transmitting a frame with the start of frame (SoF) bit that is always a dominant bit, which results in a falling edge on the CAN bus. Since multiple nodes can attempt transmitting a frame at the same time, the arbitration is resolved during the arbitration field which is different for standard (Fig. 2.1) and extended (Fig. 2.2) formats. In standard format, the arbitration field has an 11 bit ID and the remote transmission request (RTR) bit as seen in Fig. 2.1.

However, the arbitration field consists of a 29 bit ID, the Substitute Remote Request (SRR) bit, the Identifier Extension (IDE) bit and the RTR bit in extended format as seen in Fig. 2.2. The 11 bit Base ID in standard format is lengthened with 18 more Extended ID bits in extended format in order to support more CAN messages with

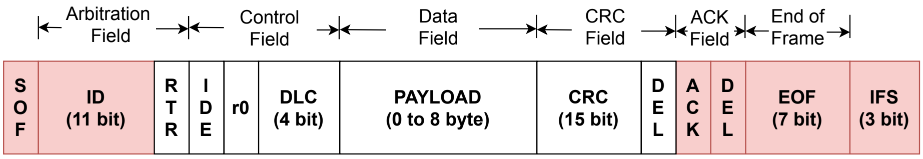

Figure 2.1: CAN frame - standard format.

unique message IDs.

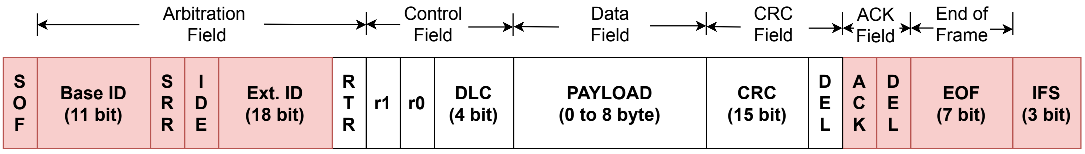

Figure 2.2: CAN frame - extended format.

RTR bit is dominant in the data frames and recessive in the remote frames. In both standard and extended formats, RTR bit is sent as the last bit of the arbitration field. SRR bit is transmitted as a recessive bit in the extended format at the same position of the RTR bit in the standard format to substitute it. Thus, an arbitration among the standard and extended frames which have the same base ID is resolved such that the standard frame continues to its transmission on the bus. In order to distinguish the standard and extended formats the value of the IDE bit is used whose position is the same in both formats. The IDE bit is dominant in the standard format and belongs to the control field. On the contrary, it is recessive and belongs to the arbitration field in the extended format.

The 6 bit control field follows the arbitration field for both standard and extended formats. In the standard format, the first bit of the control field is dominant as IDE bit whereas it is recessive as reserved R1 bit in the extended format. The following bits in the CAN frames are the same both for the extended and standard formats. A reserved bit (r0) and data length code (DLC) are sent within the control field before the data field that carries the payload of the CAN message. DLC indicates the number of bytes within the payload and the number of bytes can change from 0 to 8 bytes. The CRC field that follows the payload consists of 15 bit CRC sequence and a recessive bit as CRC delimiter. The CRC sequence is used to check the validity of the ongoing message transmission by all nodes. Afterwards, the transmitter sends a recessive bit and all receivers which receive the frame successfully send a dominant bit in the first bit of the ACK field. The second bit of the ACK field is recessive and called as ACK delimiter. Then, the transmitter continues with seven more recessive bits as the end of frame field before 3 recessive bits as inter frame spacing (IFS) that takes place before a new frame can be transmitted.

It has to be noted that during the arbitration and ACK fields, multiple nodes can access the bus by sending different bit values. However, the parts of the CAN frame where falling edges on the CAN bus can be generated only by a single node are highlighted in white in Fig. 2.2 and 2.1.

2.1.3 CAN Bit Timing

The CAN controller of each node maintains a CAN system clock. The CAN system clock signal is generally derived from a clock oscillator (CO) and its period is named as Time Quantum (TQ). Writing $f_{\mathrm{CO},N}$ and $T_{\mathrm{CO},N}$ for the nominal CO frequency and period, respectively, the nominal duration $T_{\mathrm{TQ},N}$ of a TQ is $T_{\mathrm{TQ},N} = n_{\mathrm{TQ},N} \cdot T_{\mathrm{CO},N}$.

The CAN bit timing is organized in four segments denoted as SYNC_SEG, PROP_SEG, PHASE_SEG1 and PHASE_SEG2 as illustrated in Fig. 2.3. For each CAN node N, the length of each segment is given as a number of TQs, which is fixed during run-time and whose range of values is specified in the CAN standard [4].

The TQs are counted by a TQ counter (TQC) and the SYNC_SEG consists of one TQ and PROP_SEG, PHASE_SEG1 and PHASE_SEG2 can be programmed from 1 up to 8 TQs [4]. Thus, the minimum possible length of one TQ is one twenty-fifth of the nominal bit time. Each CAN node samples the bus value at the sample point (SP) and the ratio of the SP distance to the nominal bit time is advised to be around 87.5$\%$ [38].

There are two internal bit timing synchronization mechanisms on CAN [4] within the CAN controller. In this way, the successful message transfer is provided by CAN protocol even though CAN nodes do not send and receive the bits in a synchronous way. Hard synchronization is realized at the beginning of a CAN message such that all nodes observing a falling edge when the bus is idle restart their internal bit timing

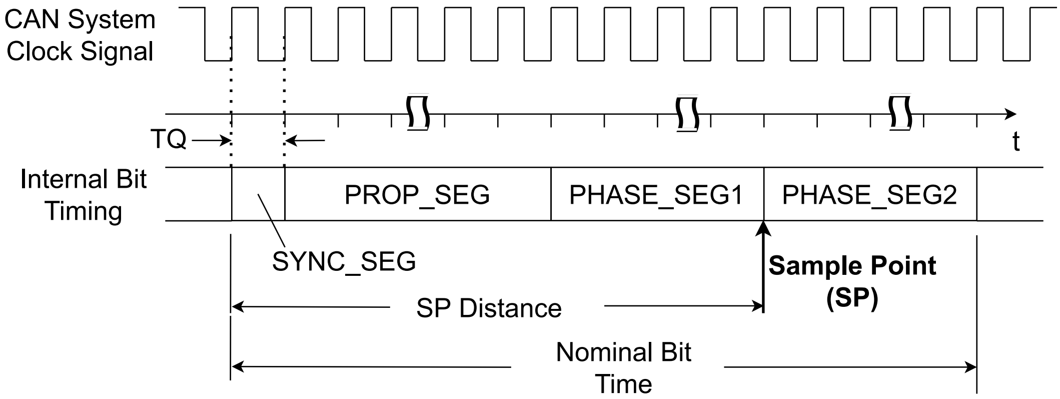

Figure 2.3: CAN bit segments.

with SYNC_SEG. That is, all nodes restart their TQC as 1 whenever they detect a SoF bit on the bus. Furthermore, the nodes apply re-synchronization during the message transmission when they receive a falling edge different from SoF bit on the bus. We write $Q_{\mathrm{F},N}$ for the value of the TQC, where CAN node N observes a falling edge and $\#_{\mathrm{TQ},N}$ for the total number of TQs in node N per one bit. Hereby, TQC counts from 1 to $\#_{\mathrm{TQ},N}$ and returns to 1 after it reaches to $\#_{\mathrm{TQ},N}$, unless an internal bit timing synchronization is performed. Then, the phase error

$$e_{\mathrm{P},N} = \begin{cases} Q_{\mathrm{F},N} - 1 & \text{if falling edge is after SYNC\_SEG} \\ Q_{\mathrm{F},N} - 1 - \#_{\mathrm{TQ},N} & \text{otherwise} \end{cases} \tag{2.1}$$

captures the deviation of $Q_{\mathrm{F},N}$ from TQC=1 (SYNC_SEG). That is, based on the value of $e_{\mathrm{P},N}$, re-synchronization shortens PHASE2 of N by $|e_{\mathrm{P},N}|$ if $e_{\mathrm{P},N} < 0$ (the CO frequency of N is slow compared to the transmitter node) and lengthens PHASE1 by $e_{\mathrm{P},N}$ if $e_{\mathrm{P},N} > 0$ (the CO frequency of N is fast compared to the transmitter node). Hereby, enough re-synchronization instants (falling edges) during a CAN message are provided by the bit stuffing mechanism, which adds a bit of the opposite value after five consecutive bits with the same value. The correction amount is limited by the configurable re-synchronization jump width (RJW). Only a node transmitting a '0' bit does not perform re-synchronization if the falling edge is observed before the sampling point [4] to prevent the application of re-synchronization with a falling edge created by itself. It has to be emphasized that the bit timing mechanism on CAN only provides synchronization relative to the SoF bit of each CAN frame and cannot directly be used for CS of LCs of different CAN nodes.

2.1.4 CAN FD Protocol

To improve the bandwidth of current CAN bus, BOSCH has introduced a new high bandwidth CAN bus called CAN FD in 2012 [16] through two major improvements: 1) increase of bit rate (more than 1 Mbps) and 2) increase of payload sizes (up to 64 bytes). CAN FD is standardized internationally in ISO 11898-1:2015 and it supports higher bit rates in a part of the data frame. Hereby, the CAN-FD controllers can also take part in standard CAN communication since the new CAN-FD frame formats make use of the reserved bits on CAN.

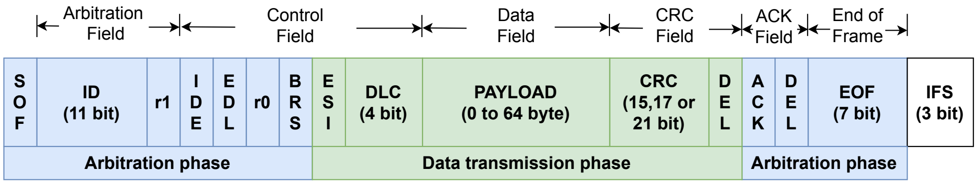

Figure 2.4: CAN-FD frame - standard format (11 bit ID).

In the standard CAN-FD frame (with 11 bit ID) as seen in Fig. 2.4, extended data length (EDL) bit is transmitted as a recessive bit after the IDE bit. Moreover, the RTR bit in CAN frame is replaced with a dominant reserved r1 bit since there are not CAN-FD remote frames. EDL bit is used to differentiate the CAN and CAN-FD frames since r0 bit in CAN frames are sent as a dominant bit unlike the EDL in CAN-FD. Then, a reserved r0 bit follows the EDL bit in the CAN-FD frame. Afterwards, bit rate switch (BRS) bit is sent. If the value of BRS is recessive, faster bit rate is applied during the data transmission phase. However, the same bit rate is used during the data transmission and arbitration phase, if its value is dominant. The duration before the BRS bit and after the CRC delimiter bit is named as the arbitration phase in CAN-FD and the same bit rates with CAN are supported. On the contrary, during the data transmission phase bit rates higher than 1 Mbps are supported by CAN-FD.

The extended CAN-FD frame is also presented in Fig. 2.5 where EDL bit is sent after reserved r1 bit which takes place instead of the RTR bit in the extended CAN frame.

The remaining bits after the EDL bit are exactly the same in standard and extended CAN-FD frames and have similar functions that are described for CAN messages in Section 2.1.2. Differently, the length of the CRC sequence can be 15, 17 or 21

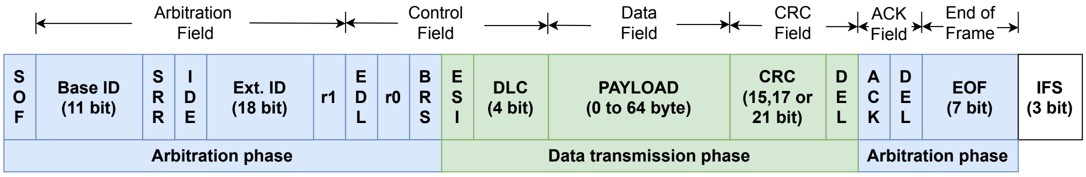

Figure 2.5: CAN-FD frame - extended format (29 bit ID).

bits in CAN-FD. Furthermore, CAN-FD frames can transmit payload up to 64bytes, different from CAN frames. Thus, the DLC values from 9 to 15 (in binary) are defined in CAN-FD for payload lengths of 12, 16, 20, 24, 32, 48 and 64 bytes respectively. The DLC values from 0 to 8 (in binary) are still represent the payloads from 0 to 8 bytes like CAN.

After installation of the nodes applying CAN-FD that enables higher bit rates, it is seen in practice that the existing high-speed CAN transceivers achieve bit rates of 2 Mbps with a linear network topology and bit rates of 5 Mbps with a point-to-point communication. Different from the existing high-speed CAN transceivers, the bit rates up to 5 Mbps is achieved with signal improvement capability (SIC) transceivers even on networks that follow star topology with long stub lines. Thus, SIC transceivers allow building more complex network topologies at higher communication bit rates since it reduces the ringing effects that are caused by reflections on a network. Therefore, the sampling of bits gets more robust when SIC transceivers are used. Moreover, CAN in Automation (CiA) has released the CiA 601-4 standard proposal specifying the ringing suppression in CAN FD star and hybrid topology networks by using CAN SIC transceivers [39].

2.1.5 CAN XL Protocol

The future of CAN might also offer interesting potentials. The next generation of CAN will be CAN XL [34] whose development process is still ongoing. This version will offer data rates more than 10 Mbps and a significantly enlarged payload of up to 2048 byte, while retaining the media access scheme of CAN and backward compatibility with the previous CAN FD version [40].

2.2 Clock Synchronization on CAN and Related Work

2.2.1 Local Clock and Clock Drift

The local time of a CAN node is commonly realized by a LC, which consists of a hardware register (HR) that is incremented using periodic ticks from a CO [41, 42]. The resolution of node N's LC is defined by the nominal frequency $f_{\mathrm{CO},N} = 1/T_{\mathrm{CO},N}$ of the CO and the number of CO ticks $n_{\mathrm{HR},N}$ per HR increment. That is, one tick of the HR corresponds to a time duration $T_N = n_{\mathrm{HR},N} \cdot T_{\mathrm{CO},N} = n_{\mathrm{HR},N}/f_{\mathrm{CO},N}$. We write $N_N(t)$ for the value of node N's HR value at time t. The maximum value of N_N depends on the width of the hardware register and hence determines the maximum time represented by the LC.

It has to be noted that the LCs of different CAN nodes are not expected to hold the same time value since two different COs cannot provide periodical ticks with exactly the same rate [41]. In particular, the actual frequency of a CO generally deviates from $f_{\mathrm{CO},N}$ due to short term, long term and environmental frequency instability effects [43, 44]. Measuring such deviations in ppm (parts per million), the frequency instability is in the order of 50 ppm for fabrication errors, 5 ppm per year for aging and 150 ppm (within -40°C and 125°C) for temperature variations [45]. That is, the actual CO frequency f_N of node N is time-varying and can be written in the form

$$f_N(t) = f_{\mathrm{CO},N} \cdot (1 + k_{\mathrm{IA},N} + k_{\mathrm{LTI},N} + k_{\mathrm{EI},N}(t)). \tag{2.2}$$

Here, $k_{\mathrm{IA},N}$ is the initial accuracy (IA) based on fabrication errors, $k_{\mathrm{LTI},N}$ is the long term frequency instability (LTI) due to aging and $k_{\mathrm{EI},N}(t)$ represents the time-varying environmental instability (EI) depending on temperature, pressure, humidity and noise from the voltage supply [41, 46, 47]. Then, the local time $c_N(t)$ of node N's LC is determined by

$$c_N(t) = N_N(t) \cdot T_N = \lfloor \int_0^t \frac{f_N(\tau)}{n_{\mathrm{HR},N}} d\tau \rfloor \cdot T_N, \tag{2.3}$$

whereby $\lfloor \bullet \rfloor$ is the floor operation. Due to the frequency instability, the LC of each CAN node diverges from real time.

We define the clock difference of node N with respect to a reference clock (RC) which is accepted to equal to the real time as

$$\Delta c_N(t) = c_N(t) - c_{\mathrm{RC}}(t). \tag{2.4}$$

When we consider that real time is given by the LC value c_{TM} of a given TM, it holds that $c_{\mathrm{RC}}(t) = c_{\mathrm{TM}}(t)$. Accordingly, the clock offset of node N in HR ticks is $\Delta N_N(t) = N_N(t) - \left\lfloor \frac{c_{\mathrm{TM}}(t)}{T_{\mathrm{N}}} \right\rfloor$. Then, the maximum value $\max_t |\Delta c_N(t)|$ describes the accuracy of the LC c_N. Furthermore, the clock drift d_N of node N is the rate of change of $\Delta c_N(t)$ and can be approximated for small values of τ as

$$d_N(t) = \frac{d}{dt}(\Delta c_N(t)) \approx \frac{\Delta c_N(t) - \Delta c_N(t-\tau)}{\tau}. \tag{2.5}$$

2.2.2 Clock Synchronization on CAN

Modern vehicles that have to be considered as safety-critical cyber-physical systems, require highly accurate CS among their distributed computing devices to synchronize actions of different nodes [48, 49, 1, 19, 20]. Since CAN is the predominant in-vehicle communication bus, it is highly relevant to support CS for CAN which does not provide a global time notion between the nodes [50]. Moreover, accurate global clocks enable more deterministic medium access and hence decrease message latencies on CAN [51, 35, 52].

Network Time Protocol (NTP) and IEEE 1588 Precision Time Protocol (PTP) [53] are the most distinguished time synchronization protocols for computer networks. The proposed protocols have been also used for different communication networks with modifications due to the differences between the network structures [54]. For example, EtherCAT utilize Distributed Clock which is similar to PTP [55]. Moreover, IEEE 802.1AS time synchronization standard [56] which sends synchronization data based on a PTP profile was specified for IEEE 802 networks. Additionally, the IEEE 802.1AS was analyzed for Ethernet based in-vehicle networks in [57]. However, the low-cost CAN cannot rely on those CS protocols due to its network structure and limited bandwidth [18, 19, 20].

Existing CS algorithms for CAN use timestamps that are taken simultaneously by all nodes on a CAN bus. A time master (TM) node (infrequently) transmits periodic RMs with its timestamp. Slave nodes perform offset correction whenever receiving a RM and optionally drift correction between RM receptions. That is, the clock accuracy of CS algorithms for CAN directly depends on the quality of the timestamps.

Without a dedicated hardware timestamping unit, the quality of the timestamps is mostly determined by inevitable software jitters until the time instant when the timer value is recorded as a timestamp. On the contrary, hardware timestamps provide more promising performance for CS applications at the expense of a dedicated hardware unit that takes timestamps with a known and constant delay. Hardware timestamping units are not very common in already deployed CAN controllers due to the lack of its standardization in the past. However, CiA 603 document which defines the hardware timestamps for CS on CAN was published in 2017 [58]. After its standardization, Bosch developed its M_CAN CAN IP with an extra hardware timestamping unit (TSU) which is able to take hardware timestamps at the SoF bit and also at the end of a frame. Moreover, there are some CAN modules [59, 60] that support hardware timestamps even before the standardization of the hardware timestamps for CS applications on CAN. However, it has to be noted that the practicability of the existing timestamping units in [59, 60] is questionable for having precise timestamps since the timestamp width is only 16 bit and the resolution can not be better than one CAN bit time.

Another important unit is the timer module for CS applications. In the cost sensitive automotive domain, the MCUs generally do not have timer modules with a property of the rate correction. Thus, the rate correction must be handled in software by adding/-subtracting 1 tick from the timer value manually. Accordingly, the timer value must be read first. Then, the new value should be calculated and should be written into the timer register. Furthermore, the time duration between the read and write functions should be compensated while deciding the new value for the timer register. Although special attention should be performed in software to compensate the time duration, the rate correction seems practicable in software with standard timer modules at the expense of an extra software load. On the contrary, some commercial off-the-shelf (COTS) MCUs [61] already provides an IEEE 1588-enabled Ethernet interface with a high precision timer module with rate correction. In the work [19], the usage of the timer's rate capability for the purpose of CS on CAN is presented.

Considering the properties of the commonly used MCUs in the automotive industry, the timer modules with rate correction and CiA 603 compatible hardware timestamping units are considered as extra hardware support in this thesis. Additionally, the CS implementations on CAN which do not require the extra hardware support is called as software-based CS (SW-CS) methods. Hereby, it has to be noted that similar classification is also followed in the literature [40, 43].

The underlying algorithm for several SW-CS methods that are based on timestamps inside the periodic RMs is proposed in [28] in 1994. It can be implemented in software without requiring any extra hardware support and uses one RM per synchronization round. Furthermore, the AUTOSAR-compliant CS method [27] that is the simplified version of PTP has been introduced for CAN. It proposes using two successive RMs as SYNC and Follow Up (FUP) to achieve better CS at the expense of the extra bandwidth consumption for CS messages. It has to be noted that the AUTOSAR CS method is also realizable on software. In a recent work [40], the AUTOSAR CS method is implemented on software with an improvement that aims to increase the software timestamping quality by using filtering. Moreover, the rate correction depending on the timestamps is also applied together with the offset correction. In this paper, the best clock accuracy that is reported experimentally is in the order of $50\,\mu$s when synchronization round is 0.1 ms.

On the contrary, the hardware based CS methods have advantages over software based CS implementations for example better timestamping quality and easiness at applying rate correction. In 1994, a special hardware mechanism is proposed by Turski [62] to take hardware timestamps at the SoF bit of a RM. Additionally, the same approach is also adopted in orthogonal clock subsystem (OCS) for CAN [43]. In order to realize those methods, the existing CAN controllers have to be modified such that they signal the instant of the SoF bit to trigger taking the timestamps. Moreover, TTCAN level 2 provides a global clock among the CAN nodes and achieves an accuracy in the order of one CAN bit time. TTCAN CS realization also depends on specific hardware units that means modifications of the CAN protocol. Additionally, in a recent work [19], the SYNC and FUP messages are used similar to the AUTOSAR CS method and rate correction is applied by using the IEEE 1588 timer with rate correction for the purpose of CS on CAN. Moreover, in the same work, a PI control loop is introduced to adjust the timer's clock rate. However, the design details about the controller is missing in the paper.

2.2.3 Gergeleit's Method

The RM transmission scheme compatible to the SW-CS method by Gergeleit [28] is illustrated in Fig. 2.6. RMs (gray boxes) with the message duration t_{RM} are sent by TM with a period T and received synchronously by any CAN node (slaves and Master) at $t_k = k \cdot T$, $k = 0, 1, \ldots$. With each RM reception at t_k, TM and each slave Si take a timestamp of their respective LC $\hat{t}_{k,\text{TM}}$ and $\hat{t}_{k,\text{Si}}$. TM then transmits $\hat{t}_{k,\text{TM}}$ with the next RM such that each slave Si receives this timestamp at $t_{k+1} = (k+1) \cdot T$. The difference between the received value $\hat{t}_{k,\text{TM}}$ and the stored value $\hat{t}_{k,\text{Si}}$ is then used to perform the clock update

$$c_{\text{Si}}(t_{k+1}^{+}) = c_{\text{Si}}(t_{k+1}^{-}) + \hat{t}_{k,\text{TM}} - \hat{t}_{k,\text{Si}}. \tag{2.6}$$

Here, $c_{\text{Si}}(t_{k+1}^{-})$ and $c_{\text{Si}}(t_{k+1}^{+})$ represent the LC value of Si before and after the update, respectively.

To this end, the application of Gergeleit's method requires the transmission of one RM in each synchronization cycle k.

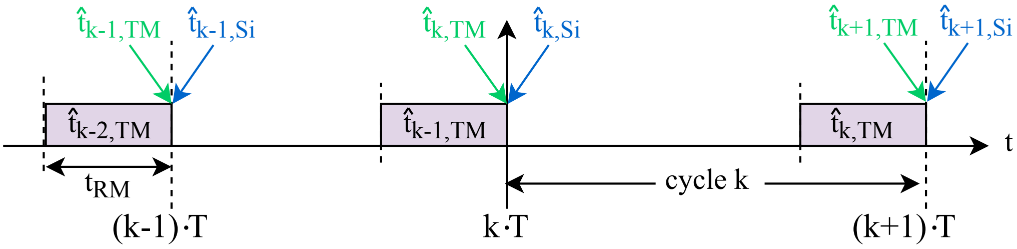

Figure 2.6: Illustration of RM transmissions compatible to Gergeleit's method.

The clock difference when the method in [28] is followed is presented in Fig. 2.7. Specifically, the figure shows the clock difference of the TM and a slave node Si. The period of RMs is T and the drift of the Si clock is d_{Si} with respect to the clock of TM. Thus, a time difference of $d_{Si} \cdot T$ is observed between two successive RMs. The blue boxes in Fig. 2.7 represent the RMs and the gray ones are regular messages transmitted on the bus.

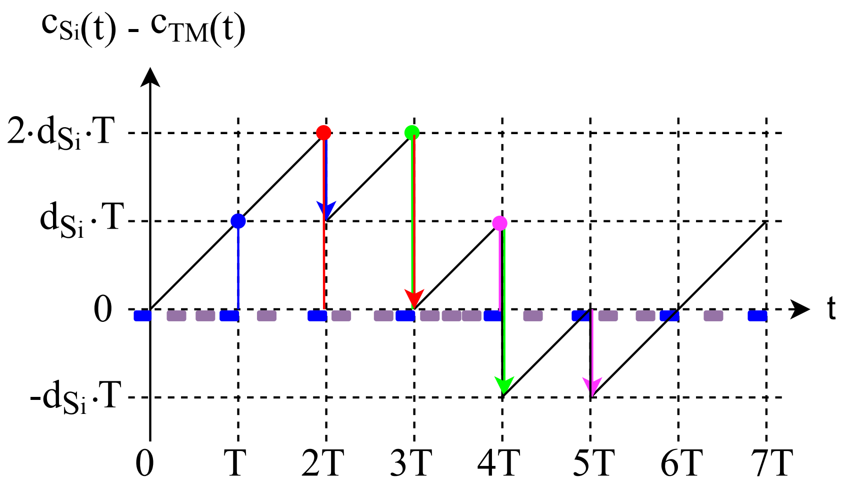

Figure 2.7: Illustration of clock difference when Gergeleit's method is followed.

For example, at time $2 \cdot T$ the slave node makes a clock correction according to the timestamp taken at time T. Since the clock difference at T is $d_{Si} \cdot T$, the slave clock value is reduced by this value at time $2 \cdot T$. It can be seen that the maximum clock difference is $d_{Si} \cdot 2 \cdot T$, which equals the theoretical upper limit guaranteed by the method in [28]. In addition, it has to be emphasized that the method in [28] cannot always fully compensate the actual clock difference right before the current RM. This is due to the additional clock drift between taking the timestamp of the previous RM and sending the timestamp in the current RM.

Additionally, there are several research works based on the method in [28] due to its low bandwidth consumption and simple implementation in software. In [63], the method in [28] is realized with a 32-bit micro-controller and it is claimed that a global time base with an approximate error of 4.5 microseconds is achieved. However, achieving clock accuracy in the order of $4.5\,\mu$s with Gergeleit's method is only possible with very frequent RMs that is not desired due to the limited bandwidth on CAN or in case of low clock drifts that is not in line with the practical usage of CAN nodes. Moreover, fault-tolerant extensions of the method in [28] for CAN are developed in [64].

2.2.4 AUTOSAR Method

According to the AUTOSAR SW-CS [27], TM sends a SYNC message right before the RM (or FUP) in each cycle k as shown in Fig. 2.8. Each slave node Si takes a timestamp $\hat{t}_{k,\text{Si}}$ when receiving the SYNC message and TM transmits its timestamp $\hat{t}_{k,\text{TM}}$ in the RM right after the SYNC message. At the reception time $t_k = k \cdot T$ of the RM, each slave corrects its LC as

$$c_{\text{Si}}(t_k^+) = c_{\text{Si}}(t_k^-) + \hat{t}_{k,\text{TM}} - \hat{t}_{k,\text{Si}}. \tag{2.7}$$

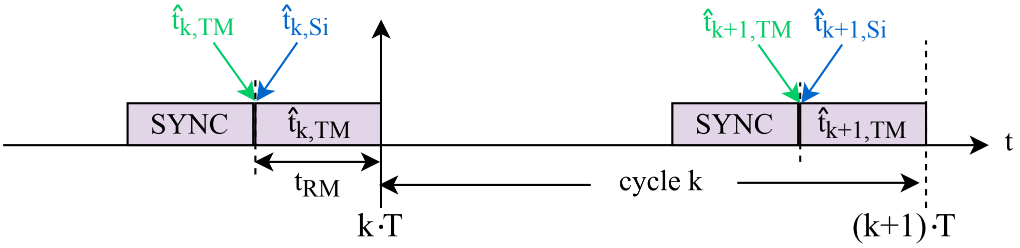

Figure 2.8: Illustration of the RMs (SYNC and FUP) according to AUTOSAR SW-CS method.

Different from Gergeleit's method, the AUTOSAR method requires two messages (SYNC and FUP) per cycle k in order to take the timestamp close to the update time.

The clock difference illustration when AUTOSAR CS method [27] is applied is presented in Fig. 2.9. The orange boxes present the SYNC messages which are sent right before the following RM (also called as FUP) in AUTOSAR CS method.

At the expense of two consecutive RMs (SYNC and FUP) in one cycle, the clock difference after the offset correction is close to the zero as desired. In the AUTOSAR method, the clock drift during the blue boxes that represent the FUP messages result in an error after the offset correction. In practice, the duration of the FUP message is very likely to be much shorter when compared to the cycle period T. Thus, the clock drift amount during the FUP message and hence the error after the offset correction is negligible with respect to the clock difference error before the offset correction that is higher than $d_{Si} \cdot T$ as seen in Fig. 2.9.

Finally, it has to be noted that the clock accuracy performance of both methods can

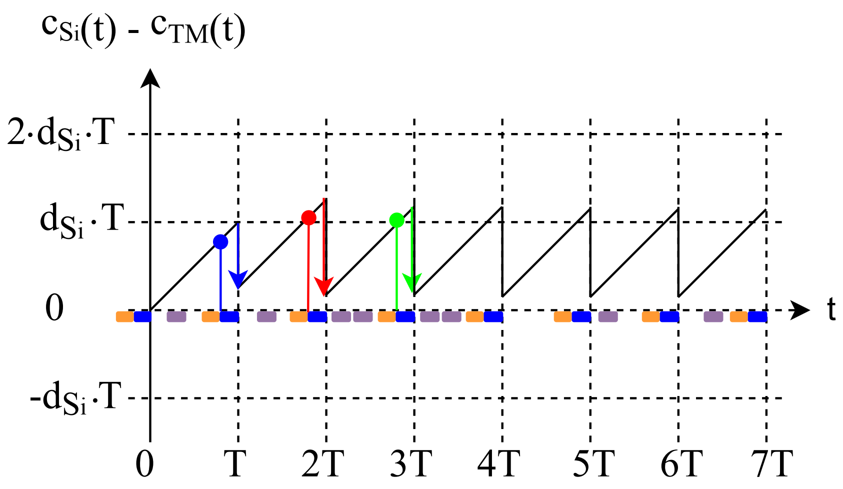

Figure 2.9: The clock difference example in case of AUTOSAR CS method.

be improved with more frequent RMs since it would result in low T values. However, this is undesired, since it would lead to a higher bandwidth usage of RMs on the already limited CAN bandwidth.

2.3 Experimental Setup

In this section, the main components of our experimental setup that is presented in Fig. 2.10 is briefly introduced. CAN nodes are realized on MCU development boards [65] and Field Programmable Gate Array (FPGA) development boards [66, 67]. Both FPGA development boards have an FPGA from Xilinx Zynq7000 SoC family that has an ARM Cortex-A9 processor additional to the programmable FPGA logic. CAN nodes communicate with each other through the CAN bus and all messages on the bus are collected on a computer via a CAN analyzer device (PCAN-USB PRO FD) which is also connected to the CAN bus. Each CAN node realizes a 32-bit LC and the LC resolution of all nodes are equal to a common network time unit (NTU) such that the clock values of different nodes are comparable [27, 26].

Regarding the clock accuracy performance evaluation, the LC values of the nodes are required to be monitored simultaneously and periodically. Therefore, all CAN nodes are triggered by a rising edge of an independent external hardware signal from a signal generator every 20 ms similar to the work in [44]. Afterwards, they send the

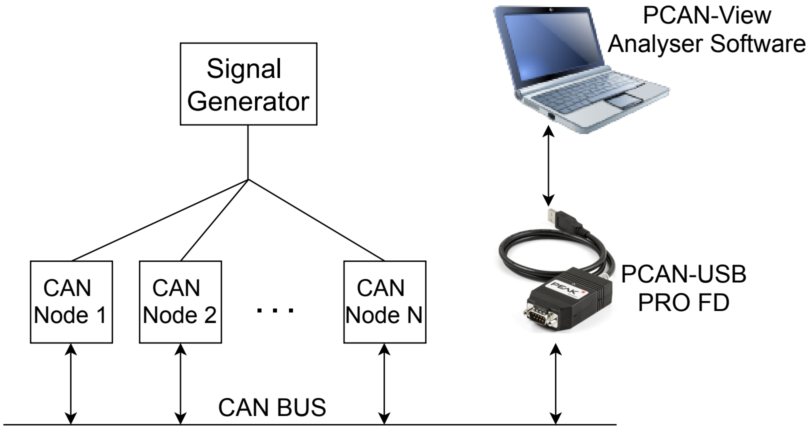

Figure 2.10: The general view of the experimental setup.

recorded LC values to the CAN bus with specific message IDs. The measurement values are collected on a computer via a CAN analyzer device and the evaluation of the measurements are realized offline.

2.3.1 Evaluation of the fundamental CS methods

This section presents the experimental evaluation of the fundamental CS methods on the setup that is introduced in Fig. 2.10. Specifically, 4 MCU boards [65] are used as CAN nodes (N1-N4). Each CAN node implements a 32-bit CS whose HR tick is arranged as 250 ns. CAN node N1 is assigned as TM whereas N2, N3 and N4 are programmed as slave nodes S1, S2 and S3, respectively. Firstly, the clock drift of S3 d_{S3} is experimentally measured as -5.8 ppm by monitoring the LCs of the nodes without applying any CS method. Afterwards, the clock difference measurements are taken for 30 minutes while Gergeleit's CS method is followed with an RM period of 1.048 ms on the bus. The clock difference measurements of S3 relative to TM are presented in Fig. 2.11 such that a small portion of them is presented as an illustration at the left side in Fig. 2.11 and the overall measurements are presented as a histogram at the right side in Fig. 2.11. Accordingly, it is seen that the measurements at the left side in Fig. 2.11 are compatible to our theoretical expectations that verifies the correct implementation of the method on the MCU boards. On the other hand, the histogram at the right side in Fig. 2.11 includes clock difference measurements that are outside of the theoretical limits that are -6 μs and 12 μs. The reason of these outliers will be explained in Section 5.2.

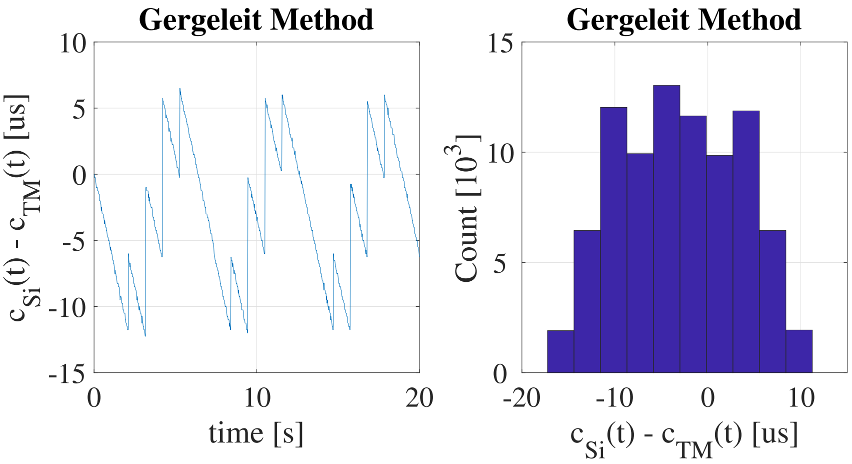

Figure 2.11: Clock difference measurement of S3: Gergeleit's method.

Furthermore, the experiment is repeated with a change that AUTOSAR CS method is followed by the CAN nodes. The representative measurements are presented in Fig. 2.12 and the clock differences are observed as between 0 μs and 6 μs, as expected.

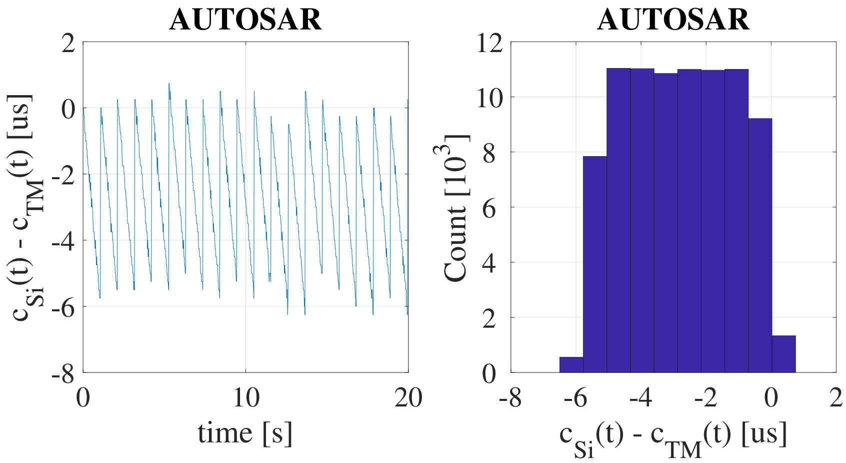

Figure 2.12: Clock difference measurement of S3: AUTOSAR.

To sum up, the practicability of the experimental setup in order to evaluate the performance of the CS methods is verified with these experiments. Moreover, it is seen that Gergeleit's method and AUTOSAR CS method are realized successfully on the development boards by comparing the experimental results to the theoretical expectations.

2.4 Timestamping on CAN and Related Work

A common shortcoming of all CAN protocols is the lack of inherent CS and security mechanisms. CS is important when coordinating tasks of different nodes in distributed real-time system applications that are supported by CAN [68, 69, 19]. Security is essential to prevent cyber-attacks in connected and autonomous vehicle applications [23]. Due to the continuing importance of CAN, the recent literature on CAN focuses on advanced CS [17, 18, 19, 20] and security methods [70, 21, 22, 23, 24, 25, 71, 72, 73, 33]. Hereby, it has to be noted that the quality of CS methods depends on accurate timestamps that are taken simultaneously by all nodes on a CAN bus. Accordingly, an AUTOSAR-compatible timestamping (TS) method is specified in CiA standard 603 [58]. Likewise, accurate timestamps are needed by timing-based IDS to improve security for CAN [74, 75, 76, 77, 78, 23].

In general, the TS process on CAN is initiated by a TS trigger signal at a certain time, denoted as the trigger (TR) instant. The TS trigger signal activates a TS service, at the end of which a copy of a timer is recorded as the timestamp. Hereby, the TR instant must occur simultaneously on all nodes on a CAN bus since any deviation leads to a deterioration of the TS quality. Regarding the TS service, it is the case that a hardware implementation can ensure a constant TS service duration [19, 43, 79], whereas the TS service duration is variable due to the inevitable software jitter in a software implementation. In summary, the quality of a TS method on CAN depends on the simultaneity of the TR instants and the constancy of the TS service duration on all CAN nodes.

CS methods for CAN commonly use timestamps that are taken simultaneously by all nodes. One node serves as a TM with a *perfect* LC and transmits periodic RMs with its timestamp. The remaining slave nodes then correct their LCs based on the timestamp in the RM and their own timestamp.

There are different choices for selecting the time instant when timestamps should be taken, which is denoted as the timestamp trigger (TR) instant in this thesis. The Gergeleit's method in [28] proposes that all nodes take timestamps at the end of message (EoM) with each RM. Conversely, timestamps are taken at the sampling point of the SoF bit of each RM in [62, 43]. These timestamping (TS) methods are as well supported recently by standard CAN controllers as specified in [80].

Moreover, there are different options for the transmission of a timestamp that is taken with a certain RM. First, it is possible that the TM transmits the timestamp in the next RM as in [28]. Second, the TM can transmit a follow-up (FUP) message right after the RM (which is denoted as SYNC message) according to the Autosar standard [27]. Third, the TM can directly put the timestamp as the payload of the ongoing RM [62, 43]. The transmission methods in [28] and [27] can be used with both of the TS choices (EoM and SoF) and are compatible with the CAN protocol specification [4]. Differently, the transmission method in [62, 43] can only be applied when taking timestamps with the SoF and does not conform to the CAN standard since the content of a CAN message has to be modified during the transmission.

It has to be noted that, although the slave nodes can correct their LCs after receiving the TM's timestamp, they will accumulate a clock difference until the reception of the TM's next timestamp due to the drift of their oscillators [43, 19, 18]. To prevent the mentioned clock drift, recent methods such as [18, 19] provide rate correction between RMs, benefiting from the existing timestamps.

It can be concluded from the previous discussion that CS methods on CAN depend on accurate timestamps taken by all CAN nodes, whereby it is expected that the quality of the timestamps directly affects the clock accuracy. Nevertheless, to the best of our knowledge, the existing literature does not provide any work that evaluates this effect.

2.5 Notation

We next introduce the notation to be employed in this thesis. We consider a CAN bus with a set of nodes $\mathcal{N}$ and a set of messages $\mathcal{M} = \{M_1, \ldots, M_m\}$ that are transmitted on the bus. For each message $M_i \in \mathcal{M}$, we define the parameters p_i, B_i, $f_i^{\max}$, $f_i^{\min}$. Hereby, p_i and B_i represent the transmission period and payload length in Byte of M_i, respectively. Moreover, we consider that CAN applies bit stuffing such that a bit of the opposite value is inserted after five consecutive bits with the same value [4]. Accordingly, the frame bit length varies between a minimum value $f_i^{\min}$ (determined by the frame header and payload) and a maximum value $f_i^{\max}$ (determined by the maximum bit stuffing) as follows [81]:

$$f_i^{\min} = g + 8 \cdot B_i + 13 \tag{2.8}$$

$$f_i^{max} = g + 8 \cdot B_i + 13 + \left\lfloor \frac{g + 8 \cdot B_i - 1}{4} \right\rfloor, \tag{2.9}$$

whereby $g = 34$ for the standard format (11-bit identifiers) and $g = 54$ for the extended format (29-bit identifiers) of CAN. The corresponding values for the possible standard format CAN frame bit lengths are shown in Table 2.1.

Table 2.1: Bounds on the frame bit length depending on B_i.

B_i	$f_i^{\min}$	$f_i^{\max}$
0	47	55
1	55	65
2	63	75
3	71	85
4	79	95
5	87	105
6	95	115
7	103	125
8	111	135

We further introduce the bit rate B and the corresponding bit time $\tau_{\text{bit}} = 1/B$. Using τ_{bit}, it is then possible to evaluate the minimum and maximum length (time duration) of M_i as $L_i^{\min} = f_i^{\min} \cdot \tau_{\text{bit}}$ and $L_i^{\max} = f_i^{\max} \cdot \tau_{\text{bit}}$, respectively. Finally, we introduce the node map $\mu : \mathcal{M} \rightarrow \mathcal{N}$ that maps each message $M_i \in \mathcal{M}$ to its transmitter node $\mu(M_i) \in \mathcal{N}$.

CHAPTER 3

THE GENERAL FRAMEWORK

The general framework of this thesis is introduced in this chapter. Firstly, the existing message scheduling approaches which are proposed for the conventional CAN protocol are presented. The conventional approaches depend on the possibility that any CAN messages may be sent at the same time. Thus, they are very pessimistic and can not provide high bandwidth utilization while trying to meet the deadline requirements also for those unlikely cases. Furthermore, the offset assignment method that can prevent the collisions from a single node is explained. Accordingly, the CAN messages that are sent from a single node can be managed such that they are transmitted with offsets. However, a global synchronization among different nodes is missing in the offset assignment methods. Despite that, the improvements in message RTs seem possible with offset assignments when compared to the conventional CAN scheduling without offsets. On the other hand, the bounded phase concept in [82] where a light CS with a clock accuracy of a few milliseconds is explained. Although the bounding phase concept achieves better RTs together with offset assignments, its performance can still be improved to reach RTs that are comparable to the message transmission times that is the ultimate reachable point in terms of a message RT. Hereby, it has to be remembered that there are several CS algorithms in the literature which are capable of providing a global clock with clock accuracies in the order of microseconds and even better. Thus, the bounding phase concept that depends on the claimed hardness of having a precise global clock does not look reasonable anymore. Afterwards, TTCAN where deterministic bus access is provided on top of CAN is explained. TTCAN requires specific hardware modifications on CAN which prevents the widespread usage of TTCAN in automotive productions. Lastly, our CANDS framework is introduced in this chapter. CANDS aims to provide the deterministic bus access on CAN without requiring specific hardware modifications, different from TTCAN. Furthermore, CANDS have a new traffic shaping strategy that is named as Weak TDMA (WTDMA) which is based on a new TDMA approach that considers the software delays and clock inaccuracies. Thus, WTDMA does not depend on an inherent CS algorithm but can perform well with any of the existing CS algorithms in the literature and can be realized in software. Moreover, the layered architecture of CANDS is presented and main responsibilities of each layers are explained in this chapter. In addition, the possible realizations for each layer are discussed by considering the existing methods in the literature together with probable improvements which will be introduced and evaluated throughout this thesis.

3.1 Message Scheduling on CAN

3.1.1 Classical Scheduling and WCRT computation

Real-time messages on CAN need to be received before their specified deadline [83, 84, 85]. To evaluate this deadline constraint, the conventional schedulability analysis proposed in [86] and its revised version [81] have been used in the automotive industry [87]. In this analysis, the worst-case response time (WCRT) of a CAN message is computed by considering the unlikely scenario, where all higher priority CAN messages are ready for transmission at the same time. Although this worst-case scenario is possible, it is very unlikely, and this pessimistic analysis commonly leads to less efficient bandwidth usage.

At that point, we need to mention about the efforts of the priority assignment methods to increase the bus utilization on CAN. The deadline monotonic and deadline minus jitter (D-J) monotonic priority assignments are claimed as optimal for CAN in [86] and [88]. However, [81] proves that they are not optimal for CAN by giving a counter example. Thus, optimal priorities for CAN messages can be determined with using the priority assignment method in [89] that guarantees to find a schedulable order if there exists. Here, it is claimed that bus utilization up to 80% can be reached with a suitable priority assignment policy [90]. Nevertheless, it is not possible to freely change the preassigned priorities of CAN messages in industrial applications

[35, 91]. Thus, the approaches for increasing the bus utilization that only benefit from the optimal priority assignments are not practical. Furthermore, such priority assignment methods provide lower bus utilization in case of tighter deadlines since the main concern of the priority assignment methods is to meet the deadline constraint that is generally assumed in the order of the message period by priority assignment methods. However, when deadlines are shorter, they will not be as successful as TDMA approaches where lower RTs are achieved thanks to the deterministic medium access.

3.1.2 Offset Scheduling

Additional to the conventional scheduling on CAN, it is shown that assigning offsets to CAN messages of individual nodes increases the efficiency of the bandwidth usage by spreading the message load of each node over time [92, 35, 82, 85]. Different from conventional methods, new RT analyses [93, 84, 94, 95] which also consider offsets have been proposed in order to compute the message RTs.

When the offset assignment is realized in different nodes independently, it is still likely that CAN messages from different nodes may be ready for transmission simultaneously. Despite that, a major performance improvement is achieved with the usage of offsets in terms of WCRTs even without applying CS [85, 35, 82].

Additionally, the bounded phases concept which benefits from CS providing a clock accuracy in the order of milliseconds has been introduced in [82]. It does not provide high clock accuracy as global clock but it is better than the case without any common clock notion. According to the evaluation results of the bounded phase notion in the paper [35], the RT of the lowest priority CAN message is improved from 20 ms to almost 9 ms with usage of offsets without having a global clock. Moreover, the same RT is seen as almost 7 ms when a light clock synchronization with an accuracy of 1 ms is provided together with the offset assignment. On the other hand, the achieved message RTs are far away from the message transmission times that must be in the order of a few hundred microseconds since the experiments are conducted with a bit time of 2 μs (500 kbps).

Furthermore, it is shown in [85] that the usage of offsets with a CS accuracy of 1 ms allows reaching an average bus load of 83% if the deadlines are assumed to be equal to message periods. It has to be noted that the given bus load is computed with the breakdown utilization notion in [90] that can to reach higher bus utilization with higher deadline requirements. Even though the bounded phase approach with offset assignments seems to facilitate CAN messages meeting their deadlines, its usability in case of tight deadline requirements is questionable. Furthermore, the advancement in CS methods that enables clock accuracies in the order of microseconds invalidates the idea behind the bounded phase notion which claims that having a precise global clock on CAN is difficult.

3.2 TDMA and TTCAN

3.2.1 TDMA

TDMA is employed in many communication systems to provide collision-free transmission with bounded access delay [36, 96, 97, 98]. In TDMA, time is divided into isolated time slots as it is presented in Fig. 3.1 and only one node is allowed to transmit in a specific time slot. Moreover, guard times, during which none of the nodes is permitted to access the network are introduced at the beginning and end of each time slot. While guard times ensure the avoidance of interference between time slots of different nodes [37, 99], they lead to a reduction in the bus utilization.

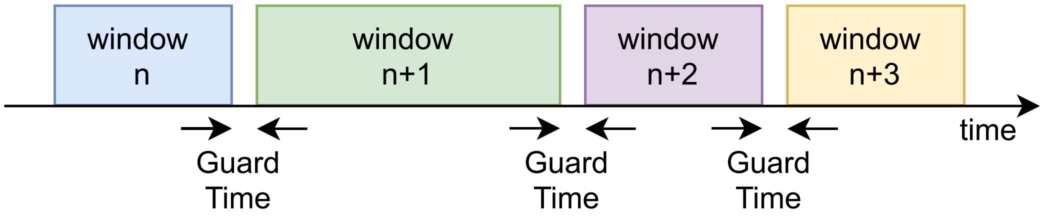

Figure 3.1: TDMA with guard times.

3.2.2 TTCAN

In addition to software-based approaches such as the offset assignment and the bounded phase concept on CAN, TTCAN [100] has been proposed to enable deterministic bus access at the expense of the modification of the CAN protocol and its underlying hardware. A bus usage close to 100% seems achievable in TTCAN since a global clock among the nodes makes it possible to schedule all CAN messages in a deterministic way [51, 35]. However, to date TTCAN has not been used in production cars [101] to the best of our knowledge due to the lack of compatibility with the existing CAN protocol and hardware.

Specifically, TTCAN Level-1 does not provide a global clock among nodes and its Basic Cycle Length (BCL) is limited as 2^{16} (65.536) bit times due to the Cycle Time counter that is 16 bits wide [100]. Although the TTCAN Level-2 implementation includes a hardware-based CS (HW-CS) method with a global clock, this comes at the expense of a hardware modification of standard CAN controllers.

The TTCAN protocol itself has several constraints in terms of schedule design. The Matrix Cycle (MC) itself is composed of a maximum number of 64 basic cycles (BC) [100]. In addition, TTCAN requires triggers that contain information about the sent and received messages and that are stored in a fixed-size memory on the TTCAN controller. Hence, the total number of triggers per node is limited [51]. Additionally, the repetition period of a message trigger has to be a power of 2 [51]. Furthermore, a CAN message is allowed to start its transmission only during the $TxEnable$ window whose length is between 1 and 16 Network Time Units (NTUs) in TTCAN [100].

Additionally, [102] provides an example of a software implementation of TTCAN that aims to work with existing standard CAN controllers by giving up the benefits of special hardware units. However, it has to be noted that this implementation is based on TTCAN Level-1 and does not support a global clock. Moreover, the experimental results are not satisfactory for the real applications in CAN networks. In [102], the message windows have a duration of 2 ms and only a bus load of 12.8% is achieved at a bit rate of 250 kbps.

3.3 Discussion

The conventional WCRT analyses for legacy CAN depend on very unlikely case for a CAN message where all higher priority CAN messages are sent together with it at the same time. Although its probability is low, it could not be ignored since it can happen, and it may result for a message to miss its deadline. Thus, the conventional CAN scheduling suffers from inefficient bandwidth usage. However, this situation can be improved when each node applies offset assignment to their own CAN messages. In case of the offset assignment, it can be avoided that the collisions between the CAN messages from the same node which may improve WCRT of the messages. Moreover, the studies in [35, 82, 85] show that the RTs of messages decrease as desired when offset assignments within each node independently from the other nodes are applied. On the contrary, it is seen from the evaluations in those papers that the low priority CAN messages still have high RTs since they can be blocked due to the higher priority messages. Moreover, even a higher priority message can still be blocked by a lower priority message since a deterministic bus access is not followed when different nodes function without having a global time notion. In order to provide completely deterministic bus access for all CAN messages independently from their priorities, TDMA should be applied that seems only possible with clock synchronization on CAN. Even though it is supported in TTCAN, the specific hardware requirements of TTCAN additional to its inherent scheduling limitations prevent it from being widely used. The deterministic bus access is highly desirable on CAN since it enables the transmission of the CAN messages without being delayed by any other message. That is, it makes possible to achieve message RTs which are determined by the message transmission times. When the deployed CAN nodes are considered, it is also required to develop a method that is able to support deterministic bus access without changing the existing hardware. Therefore, our CANDS framework is highly relevant.

3.4 CANDS Framework

In this thesis, we propose CANDS framework which aims to realize deterministic bus access on CAN, like TDMA, in order to increase the bandwidth efficiency. Although the bit rate is limited to 1 Mbps for CAN, it can meet the increasing bandwidth demands of the automotive applications if the message collisions on the bus are avoided. CANDS provides that each node transmits their CAN messages according to a TDMA schedule that is computed offline by the network designers. Moreover, deterministic bus access enables for CAN messages to reach their destinations with RTs that are only in the order of transmission times of the messages. Furthermore, a global clock among the nodes that is required to achieve a successful TDMA operation in practice is supported within CANDS with several CS method options. According to the availability of the resources, a suitable CS method can be chosen to follow within CANDS framework. The last but not the least, the classical TDMA approach is enhanced for CANDS by benefiting from the specific properties of CAN protocol.

The developed framework CANDS includes several protocol layers and components that provide improvements compared to the state-of-the-art while complying with the original CAN standard [4]. These layers include a generalized CAN controller (CC) that provides read access to additional information while ensuring the standard CAN operation; a local clock (LC) realization for a generic CAN node; a timestamping (TS) unit that allows taking precise timestamps of the LC, a CS service layer with various implementations that is able to achieve clock accuracies below 100 ns; a traffic shaping layer that follows a new WTDMA model for CAN, where time slots can temporarily overlap in practice, different from the classical TDMA. Thus, WTDMA can operate even with a moderate clock accuracy among the nodes that can even be provided with any of the existing CS methods for CAN such as [27, 18].

3.4.1 Overview

The CANDS framework consists of several layers that can be implemented in software and/or hardware. The general overview is shown in Fig. 3.2.

Hereby, it is assumed that the CAN protocol is realized on a microcontroller or FPGA hardware, denoted as the CAN system. That is, the CAN transceiver in Fig. 3.2 represents the physical connection of the CAN system to the CAN cable. The CAN Controller (CC) receives CAN messages from the upper layers and provides the corresponding physical signals to the CAN transceiver. In addition, the CC reads the

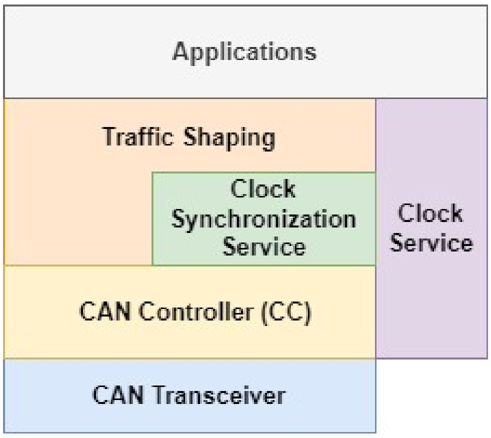

Figure 3.2: Overview of the CANDS framework.

physical signals from the CAN transceiver and provides the received CAN messages to the upper layers. The clock service implements a local clock together with a timestamping unit. It interacts with the CC to obtain trigger signals for taking timestamps. In addition, it provides timestamps and clock access to the clock synchronization (CS) service and the traffic shaping layers. The CS service performs clock updates (both offset correction and rate correction) based on information communicated among CAN nodes. Moreover, the traffic shaping layer uses the fact that clocks of different CAN nodes are synchronized in order to implement TDMA on top of the CAN protocol in order to ensure deterministic bus access with low latency.

3.4.2 Parameter Discussion

This section discusses the possible realizations for all layers and their outcomes.

The CAN controller is generally implemented on hardware. There are CAN IPs [80] for use of them inside an FPGA or there are several MCUs [59, 60, 61] that have CAN controller as a sub-module. In case of the hardware implementation, the CAN controller performs fully compatible to the CAN standard and supports the bus speeds up to 1 Mbps that is the maximum bus speed of the CAN protocol [4]. On the contrary, the work in [103] presents a software defined CAN controller (SDCC) that can run on a Cortex-M3 processor. However, the maximum bus speed is limited to 62.5 kbps when the processor runs at a clock speed of 100 Mhz. Although, the maximum bus speed is 175 kbps on the Cortex-M4F core of the LPC4357 when it runs at a clock speed of 204 Mhz. By considering the other critical tasks that should be handled in software, the achievable bus speeds will inevitably decrease with an increase in the number of other tasks different than the SDCC realization. Despite its bus speed limitation, it is very useful contribution to the literature that makes easy to realize novel ideas that requires to make modifications on the CAN controller.

The clock service unit consists of a timer module that realizes a local clock and also a timestamping unit which records a copy of the value of the local clock. The timer is generally implemented on hardware as a counter with a clock signal input. The period of the clock signal defines the resolution of the local clock. The higher clock signal frequency means better resolution that enables better performance for other layers. Together with the clock signal frequency, the width of the timer determines the overflow frequency. Although the CANDS framework handles the LC overflows, frequent LC overflows are not preferred since it will increase the code complexity of the other layers which use local clock. Moreover, the hardware resources limit the maximum clock signal frequency and the timer width. Thus, it has to be noted that increasing the clock signal frequency and the width of the timer is not practicable approach in order to have better local clock performance. Regarding the timestamping unit, it can be implemented on software where a signal comes from the CAN controller is used to trigger the interrupt service routine in the software. Within the interrupt service routine, the value of the local clock is taken as a timestamp. In the software timestamping implementation, the time duration between the trigger signal and the timestamp changes due to the software delays. Thus, the quality of the timestamps is degraded by the jitter that likely affects the performance of the other tasks benefiting from the timestamps. On the contrary, the timestamping unit can also be realized on the hardware with a constant delay that takes timestamps without a jitter. In brief, the higher quality timestamps is possible with hardware timestamping realization at the expense of a dedicated hardware TS unit.

The clock synchronization service is responsible for determining the necessary updates on the running timer that realizes the LC. It can send and receive synchronization CAN messages on the bus through CAN controller and CAN transceiver. The most important performance outcome of the CS service is the achieved clock accuracy result among the CAN nodes. Furthermore, it is also important to be friendly in terms of CPU utilization and CAN bus bandwidth consumption.

The traffic shaping layer provides a deterministic access to the bus for a CAN node. By assuming that a global clock is provided by the CS service between all nodes, a TDMA schedule where all CAN messages are transmitted in a deterministic way is achieved with CANDS framework. The traffic shaping layer monitors the local clock and transfers the corresponding CAN messages from the application layer to the CAN controller. It can be implemented on software where the software delays affect its performance while monitoring the local clock and also transferring the CAN messages to the CAN controller. Due to the software delays, the starting time instant of the corresponding message window may be missed. However, our weak TDMA model which is proposed in this thesis for CANDS considers the software delays within the traffic shaping layer additional to the clock inaccuracy among the nodes. Thus, implementing the traffic shaping layer on software is practicable. On the other hand, the hardware implementation of the traffic shaping layer makes possible to catch the message windows only with an error of the clock accuracy on the network.

Although the main goal of the CANDS framework is to increase the bandwidth efficiency and decrease the RTs of the CAN messages by enabling the deterministic bus access, the IDS and also networked control systems can benefit from the CANDS framework. A global clock and precise timestamps are required in networked control systems to provide better control performance [69]. Moreover, the timing-based intrusion detection methods can also benefit from such improvements that are provided within the CANDS framework.

CHAPTER 4

TIMESTMAPING ON CAN

Accurate timestamps are important for clock synchronization and cyber-security on the Controller Area Network. This chapter introduces a new predictable timestamping (TS) method on CAN. Different from existing TS methods, our method reduces the effect of uncertainties that are caused by the CAN bit timing, oscillator drifts and different cable lengths. Accordingly, our TS method provides an improved TS quality, which is confirmed in comprehensive hardware experiments.

The main contribution of this chapter is the development of a new predictable TS method for CAN. To this end, we first investigate the TS quality of existing TS methods that are based on the start-of-frame (SoF) bit [62, 43] and the end-of-message (EoM) [28, 58, 19]. Specifically, for the first time in the literature, we perform a detailed timing analysis of the uncertainties affecting the simultaneity of the trigger (TR) instant depending on the internal bit timing of CAN, oscillator drift between different nodes and propagation delays due to different cable lengths. In addition, we evaluate the effect of software jitter on the TS service. To mitigate the identified uncertainties of the existing methods, we propose a new TS method with a predictable TR instant. All our findings are supported by measurements from comprehensive hardware experiments. As a result of these experiments, we confirm that our new TS method provides higher quality timestamps.

To sum up, the timestamping unit of the clock service layer in CANDS framework can be designed as compatible to the existing timestamping methods that are based on the SoF and EoM, and also our predictable TS method introduced in this chapter. That is, our predictable TS method enables highly precise timestamps for the other layers inside the CANDS framework. Specifically, the performance of the clock synchronization service is expected to increase with more accurate timestamps.

4.1 Background

In order to access the physical medium, a CAN transceiver is required for a CAN controller as seen in Fig. 4.1.

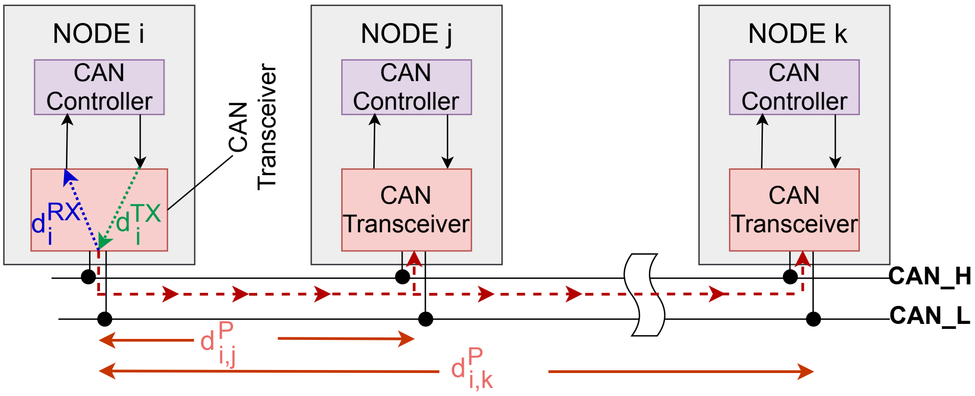

Figure 4.1: Signal propagation delays on the CAN bus.

That is, each bit of a CAN frame passes through a CAN transceiver after it is transmitted from the transmitter's CAN controller and before it reaches the receiver's CAN controller. In order to characterize the transmission path, we introduce the following notation.

- d_i^{TX}: Signal propagation delay from the CAN controller to the CAN bus through the CAN transceiver of node i.
- d_i^{RX}: Signal propagation delay from the CAN bus to the CAN controller through the CAN transceiver of node i.
- $d_{i,j}^{\mathrm{P}}$: Signal propagation delay between CAN transceivers of two node i and j.

d_i^{TX} and d_i^{RX} depend on the CAN transceiver of node i. Generally, product vendors provide upper and lower bounds for d_i^{TX} and d_i^{RX}, whose actual values can not be identical even for the products from the same vendor due to fabrication tolerances. $d_{i,j}^{\mathrm{P}}$ depends on the cable length between node i and j, whereby a delay of $5\,\mathrm{ns/m}$ over a twisted-pair cable is considered [19].

The maximum cable length depends on the physical delays and the bus speed since the signal must propagate to the farthest node and back again before each bit is sampled. It should typically be shorter than 40m [104, 19] for the maximum bus speed of 1 Mbps of the CAN protocol. In case of lower bus speeds, longer bus lengths can be used.

4.2 Components of Timestamping on CAN

Timestamping on CAN can be characterized by two main components. First, the timestamping process is initiated at the trigger (TR) instant on each CAN node. Second, the TR is detected and a copy of the current value of the local clock is taken by a function that is denoted as the timestamp service. The actual time instant when the timestamp is taken is after the completion of the timestamping service. The basic setting is illustrated in Fig. 4.2 for an arbitrary k-th timestamping instant of the TM and a generic slave node S_y.

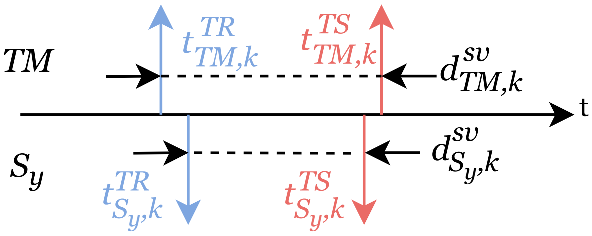

Figure 4.2: The components of the timestamping.

Here, $t^{TR}_{\mathrm{TM},k}$ and $t^{TR}_{S_y,k}$ are the respective TR instants, $d^{sv}_{\mathrm{TM},k}$ and $d^{sv}_{S_y,k}$ show the respective time intervals of the timestamping service and $t^{TS}_{\mathrm{TM},k}$ and $t^{TS}_{S_y,k}$ represent the supposedly synchronous timestamping instants for the TM and S_y.

The timestamp quality of a slave node S_y is characterized by the distribution of the differences between $t^{TS}_{\mathrm{TM},k}$ and $t^{TS}_{S_y,k}$. Accordingly, the timestamp quality directly depends on the quality of the TR instant and timestamping service. In this context, we recall that the TR is realized either by the SoF bit [62, 43] or by the end of the CAN message (EoM) [28, 18, 19, 27, 18] in the existing literature. Since both of them are directly determined by the internal bit timing mechanisms of CAN, this implies that the TR instant quality depends on the CAN protocol itself. Differently, the timestamping service quality depends on its implementation such as on software or hardware. Hereby, an implementation in hardware is expected to provide a constant time of timestamping service and will hence perform better than a software implementation with potentially varying delays.

We next determine the uncertainties of the CAN bit timing and the different timestamping methods with the aim of analyzing their effect on the TR instant and timestamping service quality.

4.3 Uncertainties Related to the CAN Bit Timing

In this section, uncertainties related to the CAN protocol are explored. We first consider that the granularity of the CAN system clock is given by one TQ. That is, even TM and S_y are aligned during the SYNC_SEG after hard synchronization or re-synchronization, the internal bit timing of the nodes can differ by the synchronization uncertainty (SYU) of up to one TQ length as illustrated in Fig. 4.3

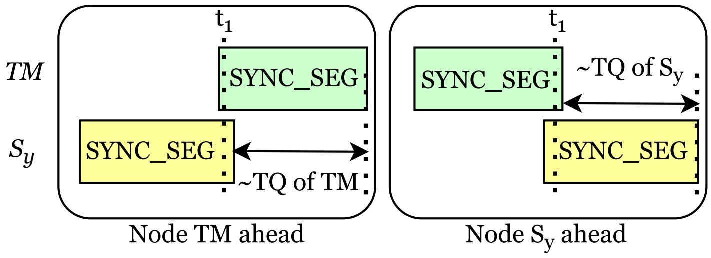

Figure 4.3: CAN bit timing synchronization uncertainty.

Remark 1 *The TQ lengths of all nodes should be chosen as small as possible to keep the bit timing SYU low.*

Remark 2 *In rare cases, the difference between internal bit timings of the nodes after re-synchronization may exceed the TQ length since the correction amount is limited by RJW.*

Second, there is a SP uncertainty (SPU) that is applicable for timestamping methods that use the SP as TR instant. Consider the case where different nodes have different internal bit timing arrangements. Then, the SP distance will be different, leading to a difference in the TR instant. For instance, the SP distance can be 70$\%$ in TM and 85$\%$ in S_y.

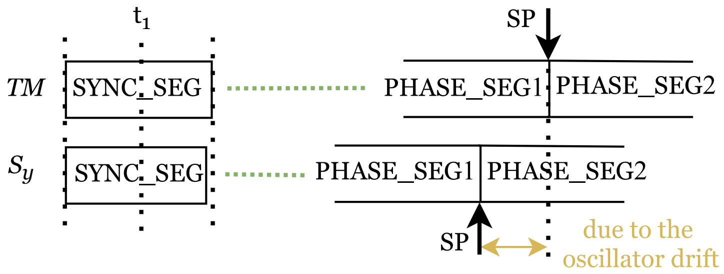

Figure 4.4: Possible time difference due to the oscillator drift.

Finally, it has to be taken into account that the actual TQ lengths of TM and S_y are different due to oscillator drift as presented in Fig. 4.4. As a result, even if TM and S_y are perfectly aligned in the SYNC_SEG and all internal bit timing parameters such as TQ lengths and number of TQs for all segments are identical, the TR instant in both nodes will be different, due to an oscillator drift uncertainty (ODU). As an example, a typical oscillator drift in the order of 100 ppm (parts per million) will result in a difference of 0.8 ns after one nominal bit time of 8 μs for a bit rate of 125 kbps. This time difference increases as long as no re-synchronization re-aligns the bit times.

Remark 3 *In order to decrease the effect of oscillator drift, it is preferable to choose the TR close to a synchronization instant.*

4.4 TR Quality using the SoF bit as TR Instant

We perform a detailed timing analysis of the components of the TR quality when using the SoF bit as TR instant. For clarity, we first consider the case in Fig. 4.5, where only the TM sends an RM, while all other nodes remain silent. Here, t_1 presents the starting instant of the SoF bit sent by the TM. That is, the CAN controller sends a bit (falling edge) at the beginning of the SYNC phase at t_1. Afterwards, the CAN transceiver starts transmitting the bit on the CAN bus at t_2 after the delay $d_{\mathrm{TM}}^{\mathrm{TX}}$ and the

CAN controller of the TM receives the bit at t_3 after $d^{\mathrm{RX}}_{\mathrm{TM}}$. Whenever TM notices the falling edge of the SoF bit, it applies hard synchronization at t_3, restarting its internal bit time with SYNC_SEG. The signal on the bus reaches the CAN transceiver of S_y after the propagation delay d^P_{TM,S_y}. Then, the CAN controller of S_y notices the falling edge of the SoF bit after $d^{\mathrm{RX}}_{S_y}$ at t_5 and also applies hard synchronization. Finally, t_6 and t_7 represent the actual TR instants of the TM and S_y.

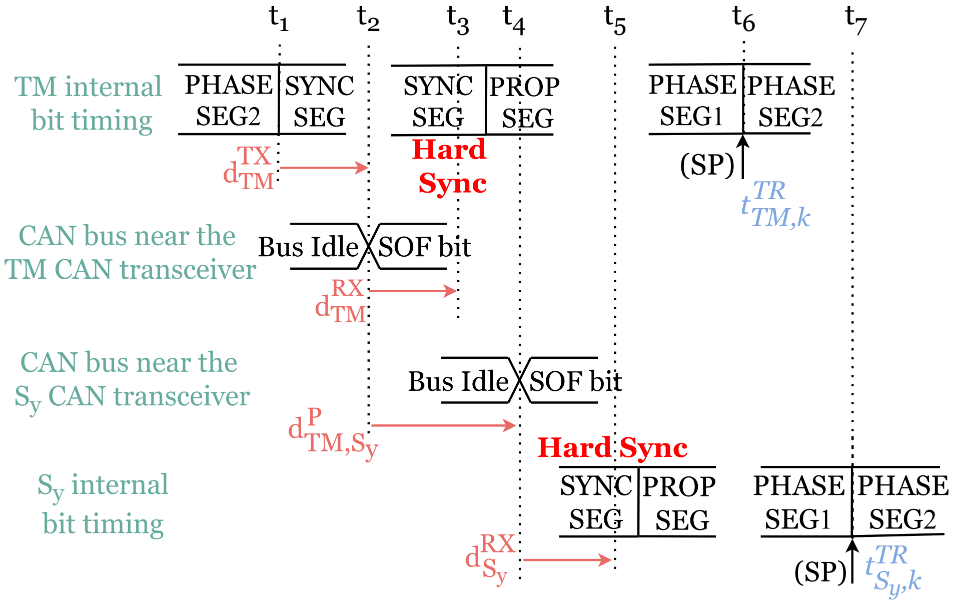

Figure 4.5: Timing analysis of SoF bit TR instant.

Looking at the components which contribute to the difference between t_6 and t_7 (and hence the uncertainty in the TR instant), d^{TX}_{TM}, $d^{\mathrm{RX}}_{\mathrm{TM}}$, $d^{\mathrm{RX}}_{S_y}$ and d^P_{TM,S_y} can assumed to be almost constant and can hence be compensated based on measurements. Accordingly, it seems that the uncertainty mostly occurs after hard synchronization and hence consists of SYU, SPU and ODU in Section 4.3 since the SP of the SoF bit is used as TR instant. Hereby, the time interval where the oscillator drift is effective is shorter than 1 bit time since the hard synchronization instant and SP take place within the same bit.

However, it has to be taken into account that the previous analysis assumes that the TM is the only transmitter on the bus. In the general case, multiple nodes may start sending the SoF bit without noticing each other due to their differences in the bit timing and because of the signal propagation delay. In this case, the relation between the synchronization instants of TM and an arbitrary slave node S_y are not directly determined by d_{TM}^{TX}, $d_{\text{TM}}^{\text{RX}}$, $d_{S_y}^{\text{RX}}$ and d_{TM,S_y}^{P} , which introduces an additional uncertainty. In this context, it is important to note that the latter case with multiple sender nodes is more likely at higher bus loads which are commonly observed in practical applications.

4.5 TR Quality using the EoM as Trigger Instant

Defining the EoM as the TR instant is a very common practice since all existing CAN controllers indicate the EoM to the upper layers to inform that message transmission or reception is complete [19, 79]. Hereby, it has to be respected that the definition of the EoM, in other words the message validation instant, is different for the transmitter and receivers according to the CAN protocol. The transmitter considers the SP of the last bit of the EOF field as the EoM, whereas the receivers have already observed the EoM at the SP of the second-last bit of the EOF field [79].

We next investigate the components of the TR quality when using the EoM as the TR instant. In principle, the timing of the EoM for each node depends on its last re-synchronization instant before the EoM. Recalling that only "1" bits are transmitted during the EOF field and the second ACK bit, this instant is either the first ACK bit or the last falling edge during the CRC field.

We first consider a slave node S_y that performs re-synchronization at t_4 in Fig. 4.6 with the last falling edge during the CRC field, whose position depends on the message content. Then, S_y accumulates ODU until the first ACK bit, where S_y transmits a "0" bit. Then, there are two possible cases for S_y depending on the distances and the uncertain relation to the bit timing of the other slave nodes in the first ACK bit. If S_y observes a falling edge due to an ACK bit that is sent by another slave node before S_y starts sending the "0" bit as the ACK bit, it performs re-synchronization. Otherwise, S_y does not re-synchronize. That is, the last synchronization instant for an arbitrary slave node S_y is not certain when the EOM is used as TR instant. Furthermore, S_y continues accumulating additional ODU until the EoM whether it performs re-synchronization or not.

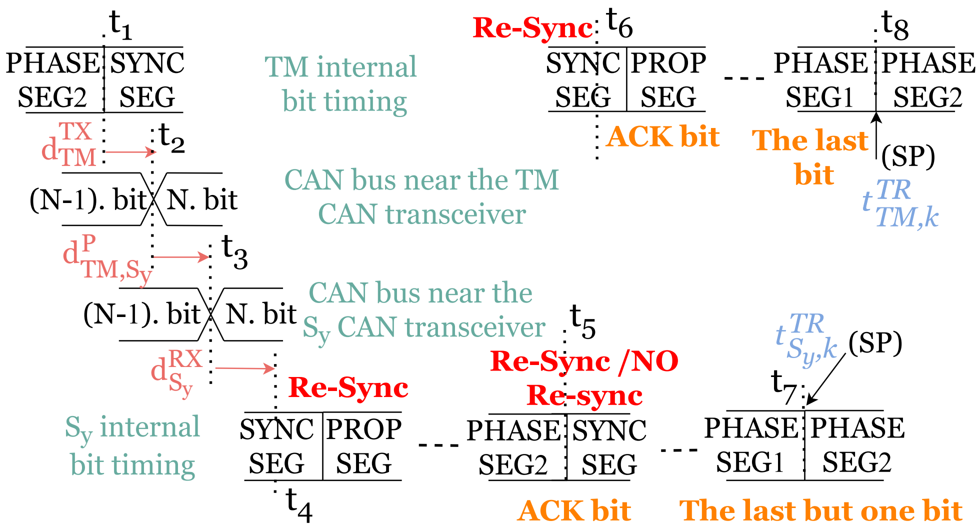

Figure 4.6: Timing analysis of EOM TR instant.

On the contrary, the TM definitely performs re-synchronization during the first ACK bit since it sends a (recessive) "1" bit on the bus, whereas all slaves send a (dominant) "0" bit. Nevertheless, it is uncertain which slave triggers the falling edge for the TM since this depends both on the bit timing of the slaves and their distance to the TM. After the first ACK bit the TM accumulates ODU until the EoM. Lastly, TR instants for TM and S_y are shown with t_7 and t_8 depend on the SPU since the SP of the last two bits of EOF field triggers the timestamping service.

Additional to SYU, SPU and ODU in Section 4.3, there are uncertainties regarding the re-synchronization instants, the nodes that trigger re-synchronization and the nodes that perform re-synchronization in the first ACK bit when using the EoM as TR instant. In addition, the duration during which ODU is accumulated is uncertain. That is, it is expected that a high TR quality cannot be ensured when EoM is used as TR instant.

4.6 Predictable TR Instant

The existing trigger options suffer from several uncertainties due to the particularities of the CAN bit timing. In order to mitigate the identified uncertainties, we propose to use a predictable TR instant. According to the previous discussion, the following properties are desired.

1. Different from the TR instant at the SoF bit or at the EoM, a falling edge that is certain to be generated by the TM should be used,
2. Different from the TR instant at the SoF bit and the EoM, the SPU and ODU should be avoided.

In order to address 1), we propose to locate the TR instant to the detection moment of the first falling edge after the CAN ID field of the frame in Fig. 2.1 and 2.2.

In this case, it is guaranteed that the falling edge is generated by the TM while transmission of an RM. We further address 2) by directly using the detection time of the falling edge as the TR instant instead of the SP, hence entirely removing SPU and ODU. That is, the uncertainty of the TR instant is limited to TQ length by SYU.

In Fig. 4.7, it is represented that the CAN controllers of TM and S_y receive the first falling edge after the CAN ID bits at t_3 and t_5 respectively. The source of the falling edge is certainly TM during the transmission of an RM in this method. By assuming that d_{TM}^{TX}, $d_{\mathrm{TM}}^{\mathrm{RX}}$, $d_{S_y}^{\mathrm{RX}}$ and d_{TM,S_y}^{P} are known, the arrival of the falling edge on an arbitrary slave node S_y and TM can be expressed formally. Therefore, $t_{S_y,k}^{TR}$ the detection instant of the falling edge on S_y can be defined relative to the detection instant on TM $t_{TM,k}^{TR}$ with only the SYU, which is bounded by one TQ of the TM or S_y.

In principle, any falling edge in white part in Fig. 2.1 and Fig. 2.2 can be used. Nonetheless, we suggest using the first falling edge after the CAN ID since it is guaranteed that this falling edge occurs before the transmission of the payload. In particular, there are 7 bits between the CAN ID and the message payload such that a falling edge is ensured because of bit stuffing. In this way, using our proposed predictable TR instant makes it possible to apply any of the options for the timestamp transmission described in Section 2.4. That is, timestamps can be transmitted in later RMs or FUP messages but can also be put in the payload of an RM during the ongoing transmission.

Remark 4 *We also note that there is no additional implementation complexity when*

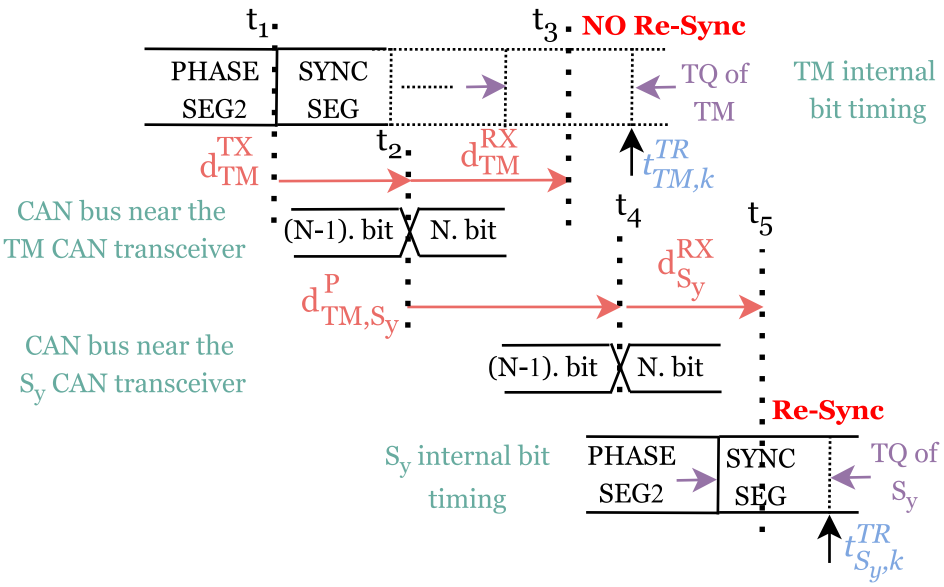

Figure 4.7: Timing analysis of the predictable TR instant.

using the proposed predictable TR instant. The already existing logic within a standard CAN controller is capable of indicating the first falling edge after the CAN ID as defined with the predictable TR instant, similar to the SoF bit and EoM indications.

Remark 5 *The previous results were discussed for the legacy CAN protocol [4]. We note that analogous results are valid for the recent extensions CAN FD and CAN XL since they use the same arbitration, bit timing and acknowledgement mechanism. Specifically, this suggests that our predictable TS method will as well be beneficial for CAN FD and CAN XL.*

4.7 Timestamping Service Quality and Discussion

As discussed in Section 4.2, the second component of the timestamping quality is the timestamping service quality that is determined by $d^{\mathrm{SV}}_{\mathrm{TM}}$ and $d^{\mathrm{SV}}_{S_y}$ of the TM and any slave S_y. On the one hand, the timestamping service can be implemented in software. This will lead to varying values of $d^{\mathrm{SV}}_{\mathrm{TM}}$ and $d^{\mathrm{SV}}_{S_y}$ due to unavoidable jitter in the software delays. Even though the jitter of the software delays can in principle be minimized by assigning higher task priorities to the timestamping service, it has to be respected that this is not applicable in practice due to other mission critical tasks of the CAN nodes. On the other hand, a hardware implementation of the timestamping service provides a constant value of $d_{\mathrm{TM}}^{\mathrm{SV}}$ and $d_{S_y}^{\mathrm{SV}}$ as desired.

Following the above explanation, it can be concluded that the timestamping service quality together with the TR instant quality defines the resultant timestamping quality when a software implementation of the timestamping service is chosen. This observation is confirmed in the experimental evaluation in Section 4.8.2. On the contrary, the quality of the TR instant directly determines the timestamping quality when using a hardware implementation. Because of this reason, we next compare the uncertainties of the different TR methods. To this end, Table 4.1 qualitatively evaluates the effect of the identified uncertainties in line with the elaborations in Section 4.4 to 4.6. Hereby, the mark ✓ indicates that the uncertainty negatively affects the quality, ✓* shows that the uncertainty negatively affects the quality but is negligible or can be compensated based on additional effort (such as measurements) and ✗ means that the uncertainty does not exist for the respective method.

Table 4.1: Dependency comparison.

	SoF	EoM	Predictable
SYU	✓	✓	✓
SPU	✓*	✓*	✗
ODU	✓*	✓	✗
CAN transceiver delays	✓	✓	✓*
Cable delays	✓	✓	✓*
Message validation instant	✗	✓*	✗

The SYU exists for all TR methods and cannot be compensated since it is caused by the granularity of the TQ. The SPU and the ODU only exist for the SoF and EoM TR method, which trigger at the SP of the respective bit. In principle, the SPU can be compensated by comparing the SP distances of the TM and the slave nodes. The effect of oscillator drifts depends on the duration between the last synchronization and the respective TR instant. For the SoF TR method, this duration is less than one bit time and hence negligible. Differently, the duration is not deterministic and can be up to 19 bit times for the EoM TR method. Furthermore, transceiver delays and cable delays affect all TR methods. However, both delays can only be compensated for the proposed predictable TR method since the TR instant directly follows from these delays as explained in Section 4.6. This is the major benefit of our method, which will be confirmed in Section 4.8.6. Finally, the EoM TR method is also affected by the difference in the message validation instant definition which is defined as the TR instant for the EoM TR method as described in Section 4.5. Hereby, the difference equals to one bit time and the effect on TR instant quality can be compensated.

In summary, it is readily observed that the EoM TR method is the least suitable method for achieving a high timestamp quality. It is further expected that the SoF TR method can perform as well as the proposed predictable TR method whenever no other node enters arbitration when the TM sends the SoF bit of an RM. In this case, it is certain that the TM causes hard synchronization of the nodes. However, this situation is not likely in practical applications with a high bus load, where multiple nodes try to access the bus simultaneously. SoF and our predictable TR instant method makes it possible to take timestamp before the payload of the ongoing RM. Finally, we recall that the SoF and predictable TR methods allow sending the timestamps within the payload of the ongoing RM unlike EoM TR method. The qualitative results discussed in this section are supported by hardware experiments in the next section.

4.8 Experimental Evaluation of Timestamping on CAN

In this section, we evaluate the performance of the different TR methods and compare the quality of the hardware and software timestamp service implementations.

4.8.1 Experimental Setup

The experimental setup consists of 4 FPGA SoC development boards [66] (TM, $S1$, $S2$ and $S3$) which have an ARM processing system and an FPGA programmable logic on the same chip. Additionally, one PCAN analyzer is used to record CAN messages on a PC as seen in Fig. 4.8.

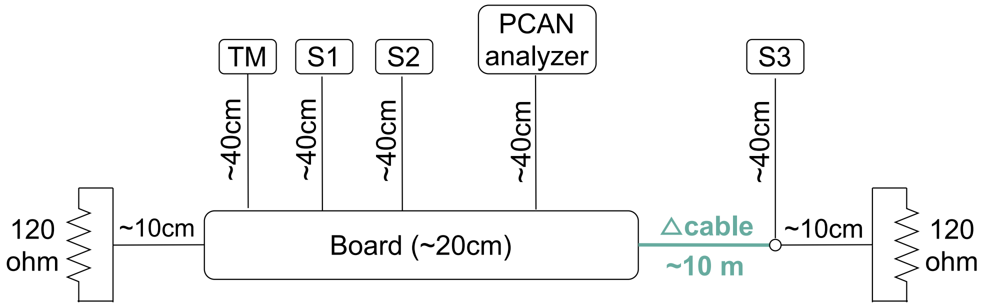

Figure 4.8: Experimental setup.

Each development board implements our custom CAN controller IP (CCIP) that is verified to successfully communicate with standard CAN controllers such as PCAN analyzer and various different COTS MCU boards in extensive experiments.

The CCIP includes a timestamping unit (TSU) that is triggered by the TS signal and that has a timer with a resolution of 10 ns. When implementing the TS service in hardware, the TR signal from the CCIP is directly used as the TS signal of the TSU. In this way, a copy of the timer value is constructed with a constant delay relative to the occurrence of the TR signal. Differently, the TR signal is connected to the interrupt input of the ARM processing system when the TS service is implemented in software. That is, the TS signal is generated whenever the ARM processing system detects the interrupt, leading to a varying delay due to the inevitable software jitter. In addition, the CCIP is developed to support the TR instants at the SoF, EoM and the predictable TR instant, whereby it has to be noted that the implementation complexity of the different TR methods is equivalent.

The overall TS quality is evaluated by observing the TS service quality and TR instant quality. In order to determine the TS service quality, each node measures the TS service delay as the difference between the TR instant and the TS signal for all timestamps with an accuracy of 10 ns. These measurements are sent on the CAN bus with specific message IDs by the nodes. Furthermore, all nodes toggle an output pin when they detect the TR instant. Those output pins are connected to another FPGA board with equal propagation delays to measure the TR instant quality of the slave nodes in real time with a timer that has 10 ns resolution.

As stated in Fig. 4.1, the TS quality is affected by the signal propagation delay. In order to study the effect for different TR methods, an extra 10 meter Δ cable is used in some experiments to increase the bus length between S3 and other nodes. Moreover, the bus load is changed with extra CAN messages which are randomly sent by all nodes on the bus.

In the subsequent sections, the TR instant quality measurements are presented with their standard deviation and peak to peak difference statistics. Both of these metrics are important and should be as low as possible. Furthermore, a consistent TR instant quality of different nodes on the bus is also an indicator of a good TR method.

4.8.2 Evaluation of the Timestamping Service Quality

In this section, the timestamp service quality is compared for hardware and software implementations. The experiments are conducted without using the extra Δ cable in Fig. 4.8. Furthermore, the CAN bus speed is $1 Mbps$ and TQ length of all nodes are 50 ns.

Firstly, the predictable TR method together with software TS service implementation is applied. Secondly, the experiment is repeated with a change that hardware TS service implementation is realized instead of software TS service. In the former one, the distribution of the timestamp service delays on different nodes are measured and shown in Fig. 4.9. It can be seen that timestamp request can be served on software with delays ranging from 600 ns to 1090 ns.

Furthermore, the distribution of the time difference between equivalent software delay measurements of slave nodes and TM is presented on the right of Fig. 4.10. Accordingly, uncertainties up to 460 ns affect the timestamp quality when timestamp service is realized on software, additional to the TR quality.

In the latter one, for hardware TS service, the timestamp quality becomes equal to the trigger quality which changes from -20 ns to 40 ns for all slave nodes as seen on the left of Fig. 4.10.

Thus, we next continue with the hardware timestamp service to compare the different trigger instant methods.

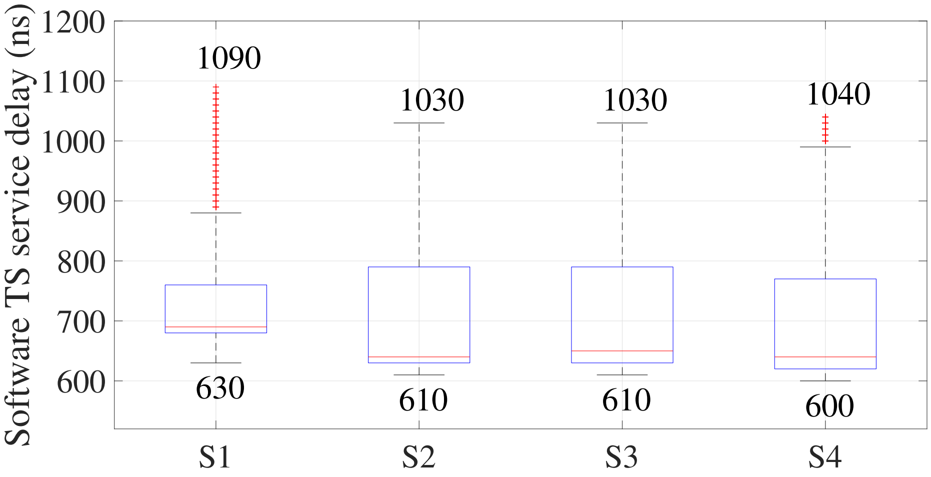

Figure 4.9: Software timestamp service delays on different nodes.

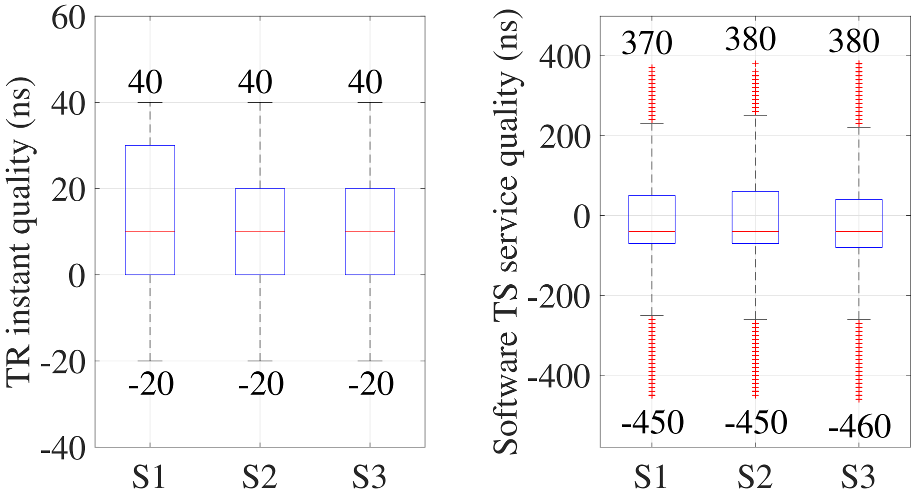

Figure 4.10: The quality measurements.

4.8.3 The Evaluation of the SoF Bit as TR Instant

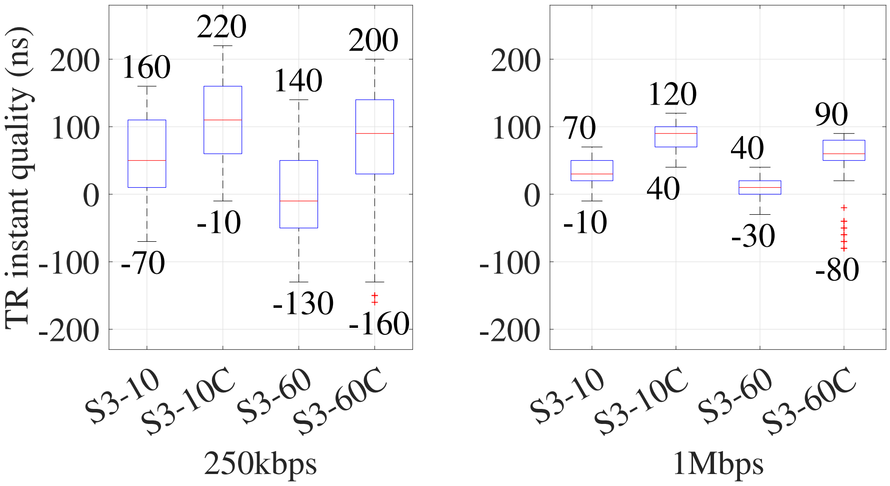

Figure 4.11: SoF bit - the trigger quality evaluation.

To evaluate the performance of using SoF bit as the timestamp trigger, another series of experiments are carried under different bus speeds, TQ lengths, cable delays and bus loads.

The TR instant quality measurements for S3 node are given in Fig. 4.11, where the right plot presents the results for the bit rate of $1Mbps$ and TQ length of 50 ns ; the left plot stands for the bit rate of $250kbps$ and TQ length of 400 ns.

Firstly, it is experimentally verified that TR instant quality is better when TQ length is smaller. The TR instant quality results at the right plot in 4.11 are small in terms of the peak to peak difference as desired since TQ length is smaller. Secondly, TR quality does not change with Δ cable delay for both bus speeds when the bus load is 10% by looking at the results $S3-10$ and $S3-10C$ where the bus usage is 10% and without and with Δ cable, respectively. The peak to peak differences are the same and the offset values of $60ns$ and $50ns$ are seen for $250kbps$ and $1Mbps$, respectively. When the signal propagation of $50ns$ due to the Δ cable and $10ns$ measurement resolution are considered, it is deduced that offset values are exactly introduced by Δ cable usage. As explained in Section 4.4, the cable delay predictably affects the TR quality for low bus loads since an arbitration with different nodes is not very likely for TM while sending the SoF bit of an RM. However, TR quality becomes worse with the usage of Δ cable when the bus load is 60% for both bus speeds according to the experiments results of $S3 - 60$ and $S3 - 60C$. Furthermore, the increase in the bus load has an effect on the TR quality according to the between $S3 - 10$ and $S3 - 60$; also between $S3 - 10C$ and $S3 - 60C$.

Although TR instant quality stays the same for the different signal propagation delays on the bus when the bus load is very low as 10%, those experiments show that using SoF bit as trigger can not guarantee a promising timestamping behaviour for practical CAN network configurations. In particular, high bus load brings more uncertainty due to the fact that other nodes likely participate in arbitration during the transmission of the SoF bit of the RM.

4.8.4 The Evaluation of the EoM as TR Instant

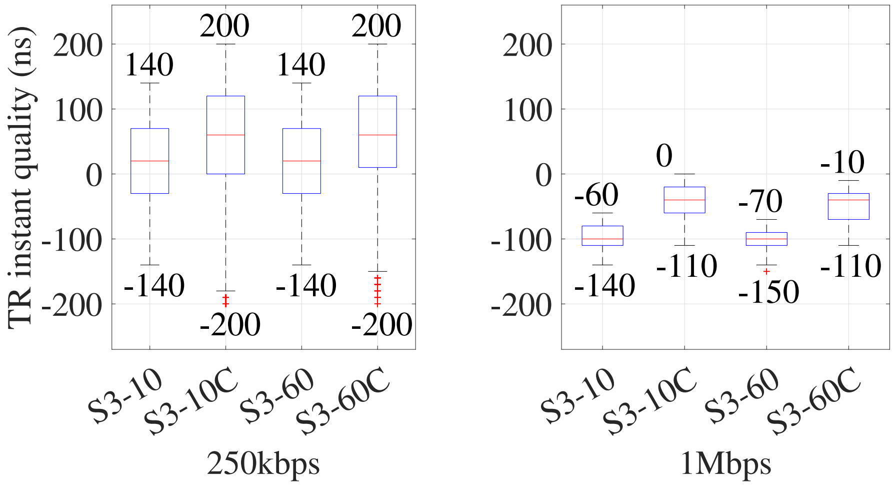

Figure 4.12: EoM - the trigger quality evaluation.

The experiments in Section 4.8.3 are repeated with a difference that EoM TR method is used instead of the SoF bit. According to the results in Fig. 4.12, the bus load has no remarkable effect on the TR instant quality when the other parameters are the same for example $S3 - 10$ and $S3 - 60$ with $250kbps$. However, the signal propagation delay due to Δ cable results in an increase in the peak to peak difference, in other words TR instant quality decreases when Δ cable is used, for example $S3-10$ and $S3 - 10C$ with $250kbps$. Unlike the SoF TR method, the signal propagation delay deteriorates the performance even for low bus loads. That is, it verifies the explanations in Section 4.5 that the uncertainties correspond to the acknowledgement bit timings when EoM is used as the trigger instant and hence are not affected by the bus load. Furthermore, it is seen from the experiments that the effect of the delays due to the network components such as cables is uncontrollable additional to SYU that depends on TQ lengths when EoM TR method is used.

4.8.5 Evaluation of the Predictable TR Instant

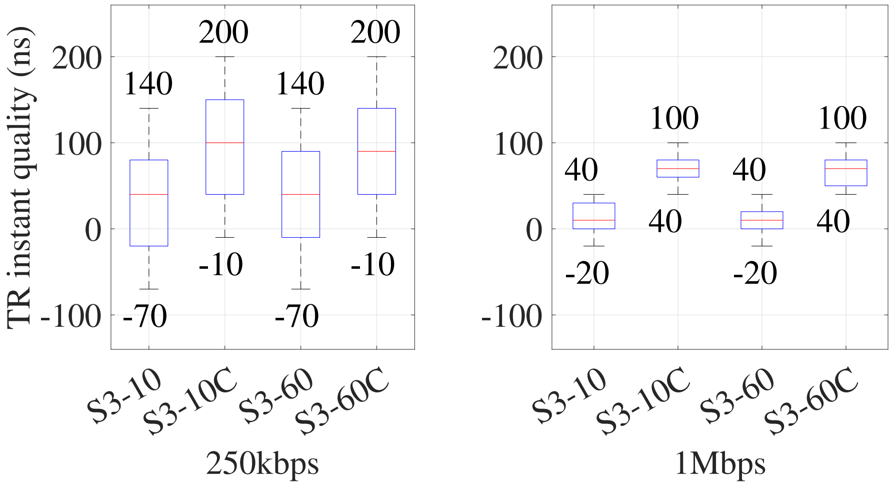

Figure 4.13: Predictable TR method - the trigger quality evaluation.

The experiments in Sections 4.8.4 and 4.8.4 are repeated with our predictable TR instant method. In Fig. 4.13, it can be seen that the peak to peak difference in other words TR quality is determined only by the bus speed. Hereby, TR quality in fact depends on the TQ length due to SYU not directly the bus speed. Different from the other existing TR methods, it becomes possible to compensate the delays only when our predictable TS method is followed since the cable delay effect is seen as a pure offset and the difference between upper and lower bounds stay the same, as seen in Fig. 4.13. To sum up, the same TR quality is preserved independently from the delays and bus load with our novel TR instant method. Thus, our predictable TR method enables the most robust trigger quality as explained in Section 4.8.5.

4.8.6 Overall Comparison

In Table 4.2 , the trigger quality measurements are also given for different bus speeds of $1 Mbps$, $500 kbps$, $250 kbps$ and $125 kbps$; TQ lengths of 50 ns, 100 ns , 200 ns and 400 ns respectively. In these experiments, the bus load is 60% and S3 has Δ cable as a challenging condition. By looking at the standard deviation and peak to peak values, the results show that our predictable TR method always provide the best trigger quality. It is also important that the trigger qualities of all slave nodes are very close to each other with our predictable TR method. However, it is seen that the SoF bit TR method can reach to the same performance with our predictable TR method for some cases. Furthermore, EoM TR method performs better than the SoF bit method only for S3 node when the bit rate is $1 Mps$ in terms of the peak to peak values. However, EoM has the worst performance in other cases. The last but not the least, it is seen that all three trigger methods perform worse when TQ length increases as explained with the timing analyses.

4.9 Discussion and Conclusion

This chapter proposes a new predictable timestamping (TS) method for the Controller Area Network (CAN), which reduces uncertainties due to the CAN bit timing, oscillator drifts as well as different CAN transceiver and cable delays. The original idea behind our method emerges from findings of a detailed timing analysis of the existing TS methods for CAN. Moreover, the results of our analysis are confirmed by comprehensive experiments that evaluate the performance of our TS method under realistic conditions. Specifically, the experiments show that our new TS method provides higher quality timestamps on CAN and is not affected by parameters that are likely to change on different CAN systems. Furthermore, our idea behind the new predictable TS method is also valid for the CAN extensions CAN FD and CAN XL since they also follow the same arbitration phases at the end and at the beginning of the CAN frames.

The clock synchronization methods provide better results with more accurate timestamps as explained in Chapter 2. Thus, our predictable TS method contributes the clock accuracy performance of the clock synchronization service in CANDS. Different from the existing TS methods that provide TS quality in the order of one bit time, our TS method directly improves the performance of the existing CS methods by providing the timestamps with TS quality in the order of one time quantum.

Table 4.2: The trigger quality comparison.

		EoM			SoF			Predictable TR		
		S1	S2	S3	S1	S2	S3	S1	S2	S3
1Mbps	mean(ns)	-105.11	-109.48	-48.60	12.83	13.02	64.39	11.20	6.3875	67.7230
	std(ns)	20.37	19.77	28.06	14.91	15.21	17.94	15.03	14.8405	14.9575
	p-p(ns)	80	80	100	60	70	170	60	60	60
500kbps	mean(ns)	-81.55	-73.00	-17.61	-9.73	-10.32	45.02	-8.24	-16.6520	39.8024
	std(ns)	42.55	42.28	55.17	29.24	29.31	30.98	29.04	29.1767	29.5149
	p-p(ns)	190	190	200	170	160	170	110	110	110
250kbps	mean(ns)	3.83	4.58	55.6116	35.03	36.03	85.33	38.44	37.4373	94.4057
	std(ns)	67.51	66.71	87.54	58.96	58.82	63.61	57.91	57.8273	58.6491
	p-p(ns)	290	280	400	270	270	360	210	210	210
125kbps	mean(ns)	65.16	63.28	86.19	-45.19	-50.86	16.04	-57.23	-56.8346	1.4020
	std(ns)	158.98	161.38	174.30	120.70	123.07	115.36	115.52	115.7222	115.2565
	p-p(ns)	790	800	780	760	730	410	410	410	410

CHAPTER 5

SOFTWARE-BASED CLOCK SYNCHRONIZATION METHODS ON CAN

The CANDS framework has a global clock among the nodes with a known accuracy. The global clock enables that the traffic shaping layers from different CAN nodes can construct a complete TDMA altogether where all CAN messages can be transmitted on the bus without any collision. That is, the global clock is one of the important building blocks that makes the deterministic bus access on CAN possible. Furthermore, the clock synchronization is also required for safety critical in-vehicle applications to coordinate their actions. Therefore, the realization of CS on CAN is highly relevant. Furthermore, developing the SW-CS methods that can run on many low-cost micro-controllers is desirable since the automotive domain is cost sensitive and hardware resources are limited.

In this chapter, the leading SW-CS methods AUTOSAR CS [27] and Gergeleit's method [28] are revisited and analyzed by considering the possible TS inaccuracies that are explained in Chapter 4. Additionally, ISCS method that can be realized on the software is proposed by mitigating the disadvantages of the existing methods. ISCS provides clock accuracy as well as AUTOSAR method while ISCS requires only one RM in a cycle, that is the half of the AUTOSAR CS method. Moreover, ISCS requires the same amount of RM bandwidth with Gergeleit's method but provides a much better clock accuracy performance than Gergeleit's method independently from the bit rate, bus load and RM period. In addition, it is shown with comprehensive hardware experiments that our ISCS method is practicable and better than AUTOSAR CS and Gergeleit's methods.

Nevertheless, ISCS, AUTOSAR CS and Gergeleit's method only perform offset correction and the clock drift continue to increase between RMs during a cycle. Therefore, RM-based drift correction CS that estimates the clock drift by using the AUTOSAR compatible timestamps is introduced in this chapter. In this method, AUTOSAR compatible offset corrections are supported by the periodic drift corrections depending on the drift estimates from the timestamps with RMs. Moreover, it is experimentally shown that RM-based drift correction CS provides clock accuracy below $4\,\mu$s by preventing the clock drift between RMs.

Different from the RM-based CS method, a controller method that also considers the previous error measurements and drift estimates is developed. The novel controller model is introduced for Gergeleit compatible timestamps first and then reconsidered also for AUTOSAR compatible timestamp scheme. In the controller-based CS methods, only periodic drift corrections are applied, and offset corrections are not required. The simulation results are validated by comparing them to the experiment results. Moreover, our detailed experiments confirm that very small clock differences below $4\,\mu$s are achieved with the controller-based CS methods.

Lastly, the dependency of the clock accuracy performance on timestamping quality is presented experimentally for CAN for the first time in the literature. A series of experiments are conducted where the bit rate is 1 Mbps and the LC resolution is 10 ns with a change in the timestamping method. It is experimentally shown that our controller-based CS method can provide CS with clock accuracies below 120 ns with high quality timestamps whereas its clock accuracy performance is measured as worse than $1.35\,\mu$s when high quality timestamps are not followed.

5.1 Revisiting the AUTOSAR Offset Correction

In this section, we revisit the AUTOSAR offset correction method that is explained in Chapter 2. The upper bound for the clock accuracy $\max_t |\Delta c_{\mathrm{Si}}(t)|$ is evaluated by considering the timestamp quality that is explained in Chapter 4. Different from the ideal case, the time difference between timestamps does not give the precise time difference between the Slave and the TM due to the misalignment of the timestamp instants.

Each slave node Si takes the current value $\hat{N}_{\mathrm{Si},k}$ of its HR as timestamp when receiving the SYNC message. Likewise, the TM takes a timestamp $\hat{t}_{\mathrm{TM},k}$ in the NTU upon reception of the SYNC message and transmits it in the FUP message right after the SYNC message. At the reception time $t_k = k \cdot T$ of the FUP message, each slave Si corrects the HR value of its LC by subtracting the measured clock offset as using $\lfloor \hat{t}_{\mathrm{TM},k}/T_{\mathrm{Si}} \rfloor$ as the TM timestamp in terms of HR ticks of Si:

$$N_{\mathrm{Si}}(t_k^+) = N_{\mathrm{Si}}(t_k^-) + \left\lfloor \frac{\hat{t}_{\mathrm{TM},k}}{T_{\mathrm{Si}}} \right\rfloor - \hat{N}_{\mathrm{Si},k}, \tag{5.1}$$

We next evaluate the clock accuracy of the AUTOSAR method, which is given by the maximum value $\max_t |\Delta c_{\mathrm{Si}}(t)|$ of the clock difference for later comparison. The CAN nodes take timestamps with a possible inaccuracy due to software delays in the intervals $[0, J_{\mathrm{TM}}]$ and $[0, J_{\mathrm{Si}}]$ for the TM and any slave Si, respectively.

In addition, it has to be taken into account that the trigger instants for timestamping of TM and Si do not occur exactly at the same instant as explained in Chapter 4. We denote this TR inaccuracy as the TRI error E_{TRI} which depends on the implemented TS method.

Finally, the LCs of TM and Si experience drift during the uncontrolled region whose duration is denoted as t_{unc} between the timestamping and offset correction instants. In AUTOSAR CS method, t_{unc} is equal to the transmission time t_{RM} of the FUP message after taking the timestamps at time $t_k - t_{\mathrm{RM}}$. Our first new result is stated in Theorem 1, which provides a bound for the clock accuracy of the AUTOSAR method. A detailed proof of Theorem 1 is given in Appendix A.1.

Theorem 1 *Consider the offset correction at time t_k according to* (5.1) *for a slave node Si with drift d_{Si}, software delays J_{TM}, J_{Si}, maximum TRI error E_{TRI} and the uncontrolled time duration $t_{unc} = t_{\mathrm{RM}}$. Then, the clock accuracy of node Si's LC is bounded by*

$$\max_t\{|\Delta c_{\mathrm{Si}}(t)|\} \leq C_{\mathrm{OC}} + \left\lceil \frac{|d_{\mathrm{Si}}| \cdot T}{T_{\mathrm{Si}}} \right\rceil \cdot T_{\mathrm{Si}}, \tag{5.2}$$

with the maximum clock difference after offset correction at the update times t_k^+

$$C_{\mathrm{OC}} = \max\{|\Delta N_{\mathrm{Si}}^{\min}|, |\Delta N_{\mathrm{Si}}^{\max}|\} \cdot T_{\mathrm{Si}} \tag{5.3}$$

and

$$\Delta N_{\text{Si}}^{\min} = \left\lfloor \frac{t_{unc} \cdot d_{\text{Si}}}{T_{\text{Si}}} \right\rfloor - \left\lceil \frac{E_{\text{TRI}} + J_{\text{Si}}}{T_{\text{Si}}/(1 + d_{\text{Si}})} \right\rceil - 1$$
$$\Delta N_{\text{Si}}^{\max} = \left\lceil \frac{t_{unc} \cdot d_{\text{Si}}}{T_{\text{Si}}} \right\rceil + \left\lceil \frac{J_{\text{TM}}}{T_{\text{Si}}} \right\rceil + \left\lceil \frac{E_{\text{TRI}}}{T_{\text{Si}}/(1 + d_{\text{Si}})} \right\rceil,$$

Hereby, $\Delta N_{\text{Si}}^{\min}$ and $\Delta N_{\text{Si}}^{\max}$ represent the minimum and maximum clock deviation in HR ticks of Si, respectively.

5.2 Revisiting the Gergeleit Offset Correction

This section evaluates the susceptibility of the Gergeleit's method to the timestamping quality. Different from the AUTOSAR method, the Gergeleit's method maintains the effects of the past errors in the following cycles since it does not use the time difference in the current cycle, as explained in Chapter 2.

In order to visualize the performance of the Gergeleit's method when ideal timestamps are not taken, an imaginary example is presented in Fig.5.1. In this example, it is imagined that the timestamps at $t = 0$ is recorded such that there is e_{TS} difference between them. Nevertheless, the difference between the recorded timestamps would be zero instead of e_{TS} in case of the ideal timestamps without an inaccuracy since the time difference between Si and TM is seen as 0 at $t = 0$ in the Fig.5.1.

Although it is imagined that the other timestamps are recorded perfectly in the following cycles with zero inaccuracy, it can be seen from Fig.5.1 that the effect of the timestamping inaccuracy at $t = 0$ is permanent in the remaining cycles.

Furthermore, the time difference likely diverges from zero when the Gergeleit's method is followed since it remembers the past errors. That is, the past errors can accumulate such that the time difference between Si and TM increases that is undesired for a CS method. We also verified this finding with series of simulations where the errors are inserted randomly at the timestamping instants as it is the case in the real-world experiments. One of the simulation results is presented as an example in Fig.5.2 where the clock accuracy exceeds the ideal upper and lower bounds which are previously presented as $2 \cdot d_{Si} \cdot T$ and $-1 \cdot d_{Si} \cdot T$ in Fig.2.7.

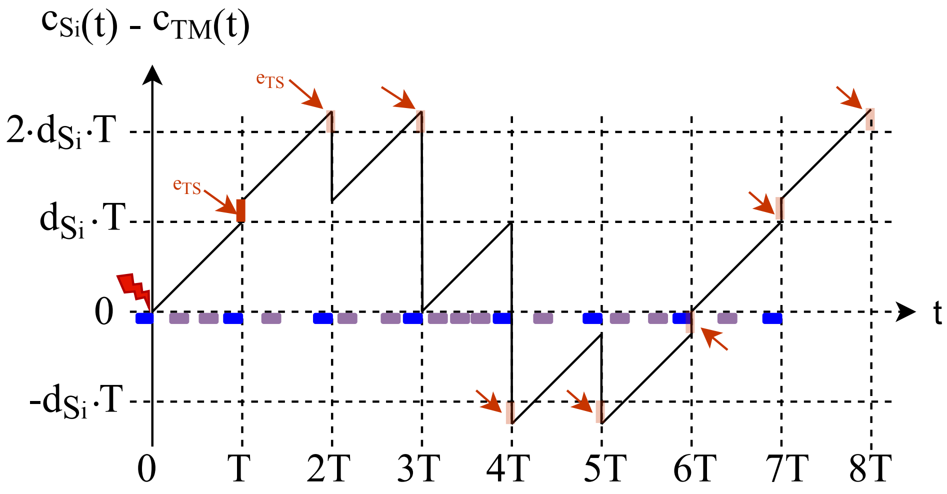

Figure 5.1: Illustration of the Gergeleit's method with a timestamping inaccuracy e_{TS}.

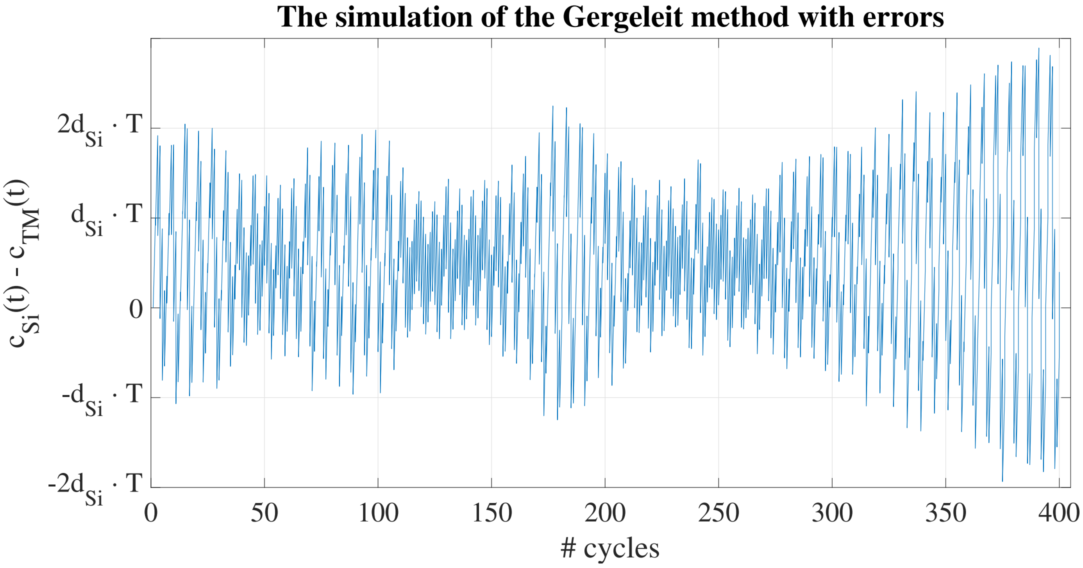

Figure 5.2: The simulation with random TS errors.

5.3 Improved Software-based Clock Synchronization Method on CAN

The new ISCS algorithm proposed in this thesis is based on two major improvements over Gergeleit's method in [28] and AUTOSAR CS method[27]. 1) The timestamps should be taken as close as possible to the offset correction instant as in AUTOSAR CS method. 2) In order to consume less bandwidth for RMs, only one RM in a cycle would be better as in [28].

In order to satisfy the improvements 1) and 2), the main idea of the ISCS method is to make use of the fact that CAN is a shared medium with message broadcast. Hence, it is possible for each CAN node to take timestamps of all messages broadcasted on the bus and to use the latest timestamp for clock correction. In this way, it is expected to achieve the same clock accuracy with AUTOSAR method with only one RM in a cycle.

5.3.1 Improved Software-based Clock Synchronization

The pseudo code of the ISCS algorithm is shown as 4 separate parts in Algorithms 1,2,3 and 4.

Algorithm 1 ISCS algorithm Part-1 / main function

```
 1: Function: main()
 2: while 1 do
 3:     if this node is a master node then
 4:         if RM period is reached then
 5:             $refTX()$; //call function $refTX$
 6:         end if
 7:     end if
 8:     if receive interrupt is intercepted then
 9:         $clockTS()$; //call function $clockService$
10:     end if
11: end while
```

Algorithm 1 constitutes the main function running on each CAN node. If the node is the TM, it must implement a timer and periodically transmit a RM whenever the timer expires. This is realized by calling the function $refTx$ in line 5. In addition, each node intercepts the receive interrupt that indicates a message reception and that is available on each CAN node. Upon detection of the receive interrupt, the function $clockTS$ is called in line 9 for applying timestamping and clock correction. The clock service on each CAN node is implemented using memory for the virtual clock, two timestamps and one Boolean flag, that indicates which of the timestamps is valid. The virtual clock is denoted as $vClock$, the timestamps are denoted as $prevTS$ (holds the previous timestamp) and $curTS$ (holds the current/most recent timestamp). The Boolean flag is denoted as $vFlag$. This flag indicates if the current timestamp or the previous timestamp is valid.

If $refTx$ is called in line 5 of Algorithm 1, the transmission of a RM with the function $refTx$ follows Algorithm 2. In this algorithm, the RM is denoted as Tx_Ref_M and has the data fields $Tx\_Ref\_M.TS$ (timestamp) and $Tx\_Ref\_M.vFlag$ (information about which timestamp is valid). Whenever $refTx$ is called, the current timestamp value $curTS$ and the validity flag $vFlag$ are written in the payload of Tx_Ref_M. Then, Tx_Ref_M is put in the Tx queue of the node.

Algorithm 2 ISCS algorithm Part-2 / transmission of a RM by TM

```
Function: $refTx$()
$Tx\_Ref\_M.TS = curTS$;
$Tx\_Ref\_M.vFlag = vFlag$;
Send $Tx\_Ref\_M$; //Put the message into the Tx queue
```

If $clockTS$ is called with the receive interrupt in line 9 of Algorithm 1, the steps in Algorithm 3 are carried out to perform timestamping and clock correction. If the received message named as Rx_M is a RM, the function $syncClock$ is called to perform clock correction in line 3. After that for all received messages, the value of the validity flag $vFlag$ is toggled (line 5), the current timestamp is written to the previous timestamp (line 6) and the current clock value $vClock$ is taken as the current timestamp (line 7).

Algorithm 3 ISCS algorithm Part-3 / clock service

```
1: Function: $clockTS$()
2: if $Rx\_M$ is a RM then
3:     $syncClock()$; //call the synchronization service
4: end if
5: toggle $vFlag$; // 0 changes to 1, 1 changes to 0
6: $prevTS = curTS$; // store old timestamp
7: $curTS = vClock$; // take new timestamp
```

The clock correction is performed by the function $syncClock$ in Algorithm 4, which is called in line 3 of Algorithm 3 whenever a RM is received. The operation of Algorithm 4 is as follows. First, the validity flags of the node ($vFlag$) and the received RM ($Rx\_M.vFlag$) are compared. In case they are identical (line 2), the clock difference of the virtual clock and the clock of the TM is determined as the difference of the timestamp in the received RM ($Rx\_M.TS$) and the current timestamp of the node (line 3). Otherwise, the clock difference is the difference of the timestamp in the received RM ($Rx\_M.TS$) and the previous timestamp (line 5). After that, timestamps and flags are reset, and the virtual clock value is corrected using the computed clock difference (line 10).

Algorithm 4 ISCS algorithm Part-4 / synchronization service

```
 1: Function: $syncClock$()
 2: if $Rx\_M.vFlag == vFlag$ then
 3:     $clockDiff = Rx\_M.TS - curTS$;
 4: else
 5:     $clockDiff = Rx\_M.TS - prevTS$;
 6: end if
 7: $prevTS = 0$; // reset previous TS
 8: $curTS = 0$; // reset current TS
 9: $vFlag = 0$; // reset validity Flag
10: $vClock = vClock + clockDiff$; // Local Clock offset correction
```

In Algorithm 4, the lines from 2 to 6 are critical for the correct operation of the proposed algorithm. In order to explain this part of the algorithm, we first state two basic assumptions. First, we assume that all nodes that want to synchronize with the TM know the CAN ID of the RM. This is a basic assumption for all clock synchronization algorithms on CAN, which is needed to detect the reception of a RM in line 2 of Algorithm 3. Second, it is assumed that the RM has the highest priority on the CAN bus. We next describe the importance of this requirement.

We first note that all nodes perform the same update after regular messages and also after receiving RMs. That is, upon reception of a message all nodes have the same value of $vFlag$. If the TM transmits a RM, it puts its value of $vFlag$ and its current timestamp in the RM. Since the clock synchronization algorithm is realized in software, it holds that Tx_Ref_M does not enter the CAN arbitration immediately when it is generated in Algorithm 2. That is, even Tx_Ref_M has the highest priority it is possible that another message, say $otherM$, is already being transmitted on the CAN bus when Tx_Ref_M enters the Tx queue on the TM. Since CAN is non-preemptive, this means the transmission of Tx_Ref_M has to wait until the transmission of $otherM$ is completed. In this case, all nodes receive $otherM$ and change their value of $vFlag$ as well as $prevTS$ and $curTS$. That is, when Tx_Ref_M is received by the nodes as Rx_M, $Rx\_M.vFlag$ is different from the value of $vFlag$, and now $Rx\_M.TS$ corresponds to the timestamp in $prevTS$ in all nodes. This situation is addressed in line 5 of Algorithm 4. In the case that Tx_Ref_M can be transmitted without interference from another message, $Rx\_M.TS$ and $Rx\_M.vFlag$ correspond to the $curTS$ and $vFlag$ in all nodes such that line 3 in Algorithm 4 is used.

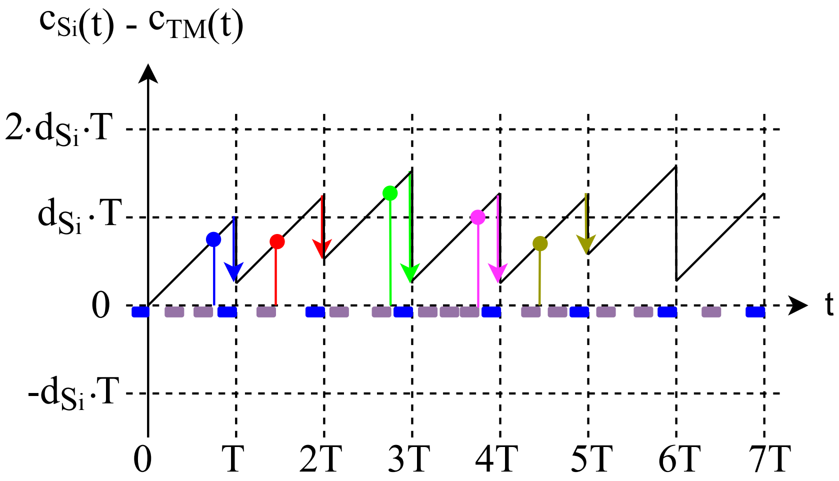

Figure 5.3: Illustration of ISCS method.

Fig.5.3 illustrates the ISCS method. According to the proposed algorithm, the slave node performs a clock correction whenever a RM is received. The amount of the clock correction equals the clock difference at the reception time of the previous message on the CAN bus as long as no other message blocks the RM on the bus (line 3 in Algorithm 4). In Fig.5.3, this situation is for example observed at the times T, $2 \cdot T$, $3 \cdot T$ and $4 \cdot T$. The situation is different at time $5 \cdot T$. Here, another message blocks the immediate transmission of the RM. That is, according to line 5 in Algorithm 4), the clock correction is done using the clock difference of the message before.

We note that the performance of the clock correction directly depends on the time difference between the reception of the message before the RM and the RM. If the last received message is close to the RM, the clock correction is very accurate. This is always the case on CAN with a high bus load. We also note that the clock difference does not increase cumulatively. For example, at time $2 \cdot T$, the algorithm cannot compensate the clock difference very well since the last received message is not very close to the RM. However, this case does not affect the performance of the following cycles. At the time $3 \cdot T$, the last message is very close to the RM. Thus, most of the clock difference is compensated after receiving the RM. This independence of the successive cycles contributes to the robustness of the algorithm. The effectiveness of the algorithm can as well be seen when looking at the clock difference values right after receiving the RMs. Although, the possible upper limit of the clock difference is the same as for Gergeleit's method in [28] (that is, $2 \cdot d_{Si} \cdot T$), this case is very unlikely in the practical case where many messages are transmitted between RMs. In terms of resource usage, ISCS algorithm requires one more timestamp register and one-bit flag register when it is compared with Gergeleit's method [28]. Moreover, timestamps are taken with each message instead of each RM. Considering the prospective benefits, this amount of increase in the processor usage is acceptable. Furthermore, ISCS method consumes half of the RM bandwidth when it is compared to the AUTOSAR CS method [27]. However, it is very likely that ISCS method performs as well as AUTOSAR CS method in terms of the clock accuracy in practice. To sum up, ISCS method has remarkable advantages over the leading CS methods [27] and [28].

5.3.2 Analysis of the ISCS Method

In this section, the clock accuracy bounds for ISCS method are discussed as similar to AUTOSAR CS and Gergeleit's method.

Remark 6 *The Theorem 1 that gives the clock accuracy bounds for AUTOSAR can also be followed for ISCS method. In case of ISCS, the uncontrolled time duration* t_{unc} *depends on the regular CAN message traffic on the bus. Although* t_{unc} *can be as high as one RM period T for ISCS unless there is not any CAN message traffic on the bus between RMs, it is very likely that* t_{unc} *can be as low as* t_{RM} *when there is a CAN* *message right before the RM. In this case, ISCS provides the same clock accuracy performance with AUTOSAR method.*

It seems possible to define an upper bound for t_{unc} and hence an upper bound for the clock accuracy of ISCS method when the periods of periodic CAN messages are given. That is, by considering a periodic CAN message $M_i \in \mathcal{M}$, the minimum number of repetitions of M_i between two successive RMs with a period of T equals to $\left\lfloor \frac{T}{p_i} \right\rfloor$. When there are m periodic CAN messages ($M_1, ..., M_m$), the total transmission of those m CAN messages take at least T_{occ}^{min} between two successive RMs. That is, the minimum occupied time duration T_{occ}^{min} can be formulated as :

$$T_{occ}^{min} = \sum_{i=1}^{m} \left\lfloor \frac{T}{p_i} \right\rfloor \cdot L_i^{min}. \tag{5.4}$$

In the worst case for ISCS method, t_{unc} equals to $T - T_{occ}^{min}$ when all periodic CAN messages are sent successively at the beginning of the cycle without any idle time between them and also the message lengths are assumed to equal to their minimum lengths as L_i^{min}. In brief, after calculating the upper bound of t_{unc} according to the given periodic CAN messages, the Theorem 1 can be applied in order to determine the bounds of the clock accuracy of ISCS method.

5.3.3 Evaluation of the ISCS Method

This section evaluates the ISCS method together with the corresponding software-based CS methods: Gergeleit's CS method and AUTOSAR CS method.

5.3.3.1 Experimental Setup

The experimental setup consists of 4 MCU boards [65] as CAN nodes TM, S1, S2 and S3. Each CAN node realizes a 32-bit local clock by using the timer modules inside the MCU. The timer modules run at the speed of 4 Mhz and hence NTU equals to 250 ns.

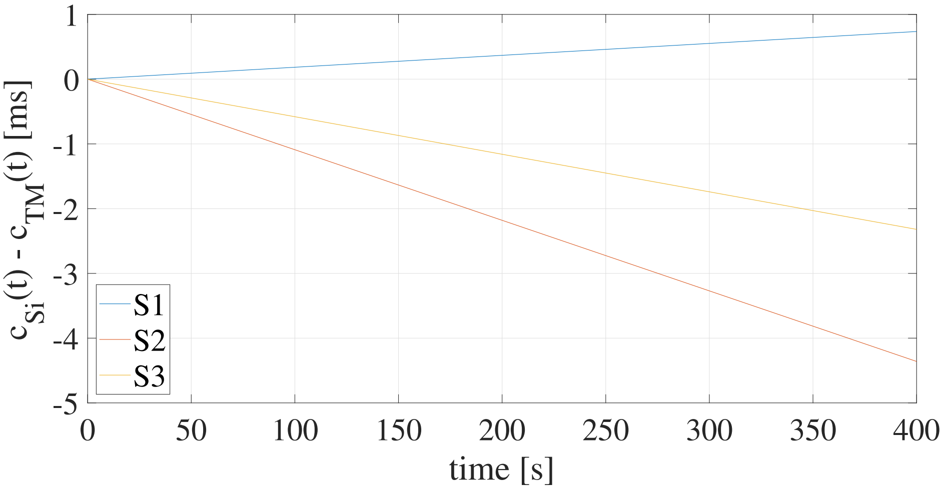

Figure 5.4: Clock drift.

Moreover, Fig.5.4 shows the clock drift of the nodes S1, S2 and S3 relative to TM in this experiment when a CS method is not followed. Accordingly, the clock drifts are measured as 1.84$\mu s/s$, -10.9$\mu s/s$ and -5.8$\mu s/s$ for S1, S2 and S3, respectively.

5.3.3.2 Experimental Results

In the Fig.5.5, the performances of the algorithms in different RM periods are presented. In the experiments, the bus speed was chosen to be 125 kbps and the bus load to be 70%. As it is expected, the mean clock accuracy value decreases in the AUTOSAR when RMs are sent more frequently with shorter RM periods. Furthermore, it is seen that our ISCS method performs as well as AUTOSAR method in practice, as it is explained. On the other hand, the performance of the Gergeleit's method also increases in case of more frequent RMs even though its performance is worse than the other methods in all cases.

Regarding AUTOSAR CS and ISCS, it should be noted that the maximum clock difference measurements show us that the improvement when changing T from 1 s to 0.5 s , could not be reached when T changes from 0.5 s to 0.25 s. Although the mean clock difference values show linear improvements according to the RM period, the maximum clock difference seems saturated due to the fact that TS quality becomes more significant when it is compared to the offset correction amount. Thus, trying to achieve better clock accuracy by reducing the RM period does not always work in practice due to the TS inaccuracies.

Regarding Gergeleit's method, the theoretical approaches lead us that the maximum clock accuracy value is almost twice of the clock accuracy in AUTOSAR CS. In our experiments, the expected theoretical clock accuracy values for S3 are 11.6 μs and 5.8 μs when $T = 1$ s for Gergeleit's method and AUTOSAR CS, respectively. However, Fig.5.5 validates that the Gergeleit's method performs much worse than its theoretical expectation since it can accumulate the past errors as explained in Section 5.2.

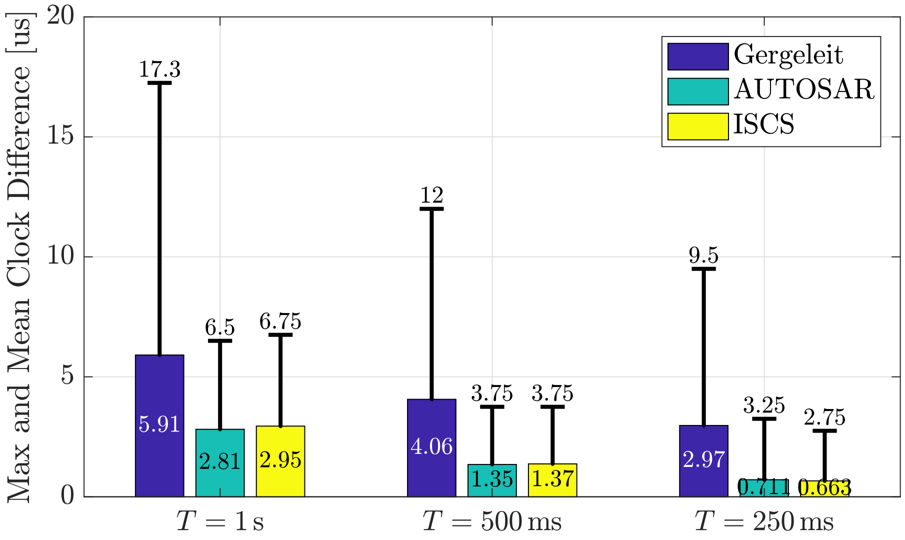

Figure 5.5: The dependency of the clock accuracy of S3 on the RM period.

We next compare the SW-CS methods under different bus loads while the bit rate is chosen as 125 kbps and $T = 1$ s. The performance of the Gergeleit's method does not depend on the other messages on the bus. However, the clock accuracy results for Gergeleit's method in Fig.5.6 changes for the different bus loads indeterminately due to nature of the Gergeleit's method that does not guarantee an upper bound in practice. Nevertheless, a slight decrease in the performance of our ISCS method can be observed as it is expected since the probability of having a regular message close to the RM is lower in case of low bus load. However, it can be seen from Fig.5.6 that our ISCS performs very close to the AUTOSAR CS method even when the bus load is 20%.

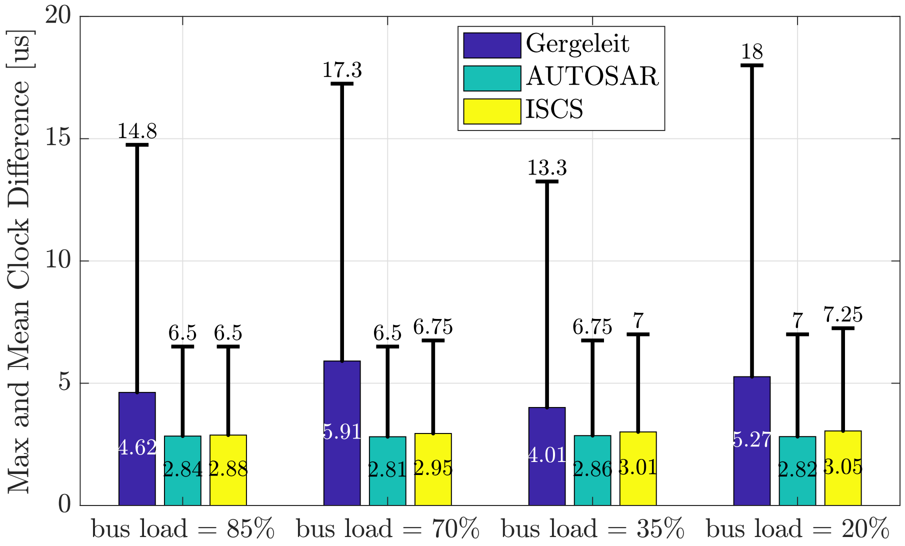

Figure 5.6: The dependency of the clock accuracy of S3 on the bus load.

In Fig.5.7, their performances with different bit rates as 125 kbps, 500 kbps and 1 Mbps are compared while the bus load is 70% and $T = 1$ s. As a consequence, it is seen that ISCS and AUTOSAR performs better than Gergeleit's method independently from the bit rate.

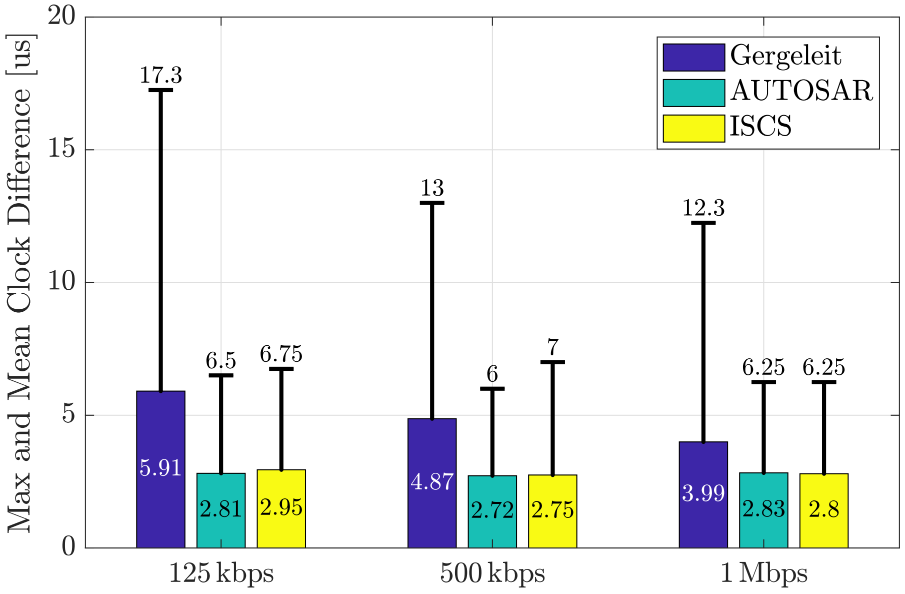

Figure 5.7: The dependency of the clock accuracy of S3 on the bit rate.

The clock difference measurements are presented in Fig.5.8 when $T = 1$ s, the bus load equals to 70% and bit rate is 125 kbps. The clock difference measurements are recorded for 10 minutes and only 20 seconds portion of the measurements are presented to provide better visibility. The results in Fig.5.8 confirms that each CS method is implemented on the CAN nodes successfully.

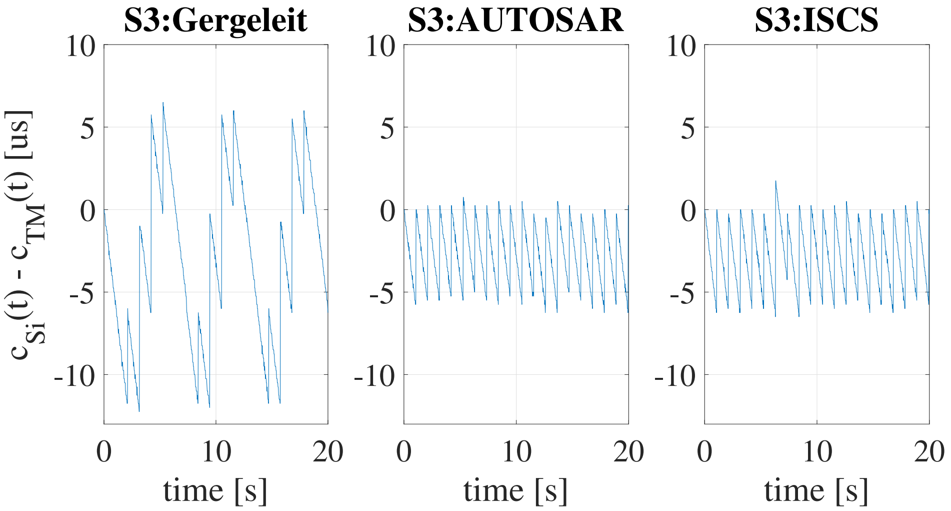

Figure 5.8: The clock accuracy of S3 for different CS methods.

5.4 RM-based Drift Correction Clock Synchronization for the Controller Area Network

In this section, a drift estimation approach which uses the periodic timestamps that are taken with RMs is introduced. Furthermore, the CS method which applies offset correction as described in AUTOSAR [27] together with the drift correction according to the estimated drift values is evaluated experimentally.

5.4.1 Drift Estimation based on RMs

The clock accuracy of CS methods can be improved using drift correction by regularly updating the LC. Hereby, we note that LC updates should only be performed in multiples of the HR tick T_{Si}. We recall the clock drift d_{Si} of Si and write s_{Si} for the sign of d_{Si}. Now consider that the LCs of TM and Si are perfectly aligned at some time t_0 such that $\Delta c_{\text{Si}}(t_0) = N_{\text{Si}}(t_0) \cdot T_{\text{Si}} - N_{\text{TM}}(t_0) \cdot T_{\text{TM}} = 0$. Then, it is only reasonable to perform LC updates at times t_1 where $(N_{\text{Si}}(t_1) \cdot T_{\text{Si}} - N_{\text{TM}}(t_1) \cdot T_{\text{TM}})/T_{\text{Si}} \approx s_{\text{Si}} = \pm 1$

as illustrated in Fig. 5.9 for a positive and negative clock drift. Moreover, it holds that a deviation of s_{Si} HR ticks on Si is accumulated within the time $s_{\text{Si}} \cdot T_{\text{Si}}/d_{\text{Si}}$.

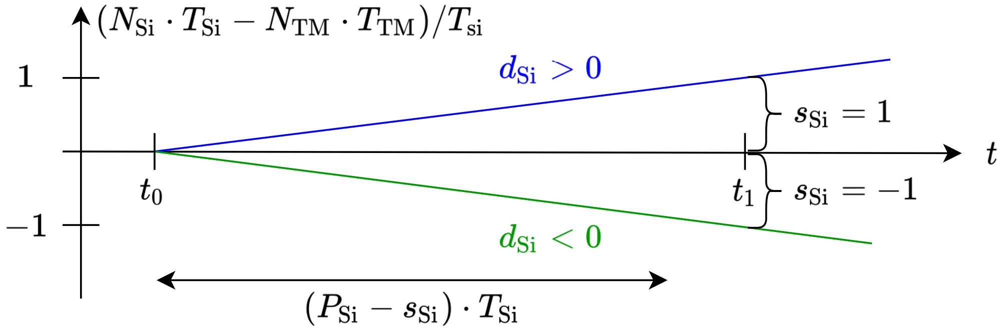
$(N_{\text{Si}} \cdot T_{\text{Si}} - N_{\text{TM}} \cdot T_{\text{TM}})/T_{\text{si}}$
1
-1
$d_{\text{Si}} > 0$
$s_{\text{Si}} = 1$
t
t_0
t_1
$s_{\text{Si}} = -1$
$d_{\text{Si}} < 0$
$(P_{\text{Si}} - s_{\text{Si}}) \cdot T_{\text{Si}}$

Figure 5.9: Computation of the LC update interval.

Writing P_{Si} for the number of HR ticks of Si's LC during the time $s_{\text{Si}} \cdot T_{\text{Si}}/d_{\text{Si}}$ (with drift), this implies that drift correction can be applied by adding $-s_{\text{Si}}$ HR ticks to N_{Si} every P_{Si} HR ticks. It has to be noted that this procedure might lead to very frequent clock updates depending on the value of P_{Si}. In order to trade off clock accuracy and processor load, it is further possible to scale the update interval by an integer γ such that an update of $-\gamma \cdot s_{\text{si}}$ HR ticks is performed every $\gamma \cdot P_{\text{si}}$ HR ticks.

In order to find P_{Si}, we first note that the correct number of HR ticks (without drift) during the time $s_{\text{Si}} \cdot T_{\text{Si}}/d_{\text{Si}}$ should be $P_{\text{Si}} - s_{\text{Si}}$. Computing P_{Si} as integer, we get

$$(P_{\text{Si}} - s_{\text{Si}}) \cdot T_{\text{Si}} = \frac{s_{\text{Si}} \cdot T_{\text{Si}}}{d_{\text{Si}}} \Rightarrow P_{\text{Si}} = \left\lceil s_{\text{Si}} + \frac{s_{\text{Si}}}{d_{\text{Si}}} \right\rceil. \tag{5.5}$$

With (5.5), P_{Si} can be determined from an estimate of the clock drift d_{Si}. Using the information from RMs, we introduce AUTOSAR-DC as an extension of the AUTOSAR method with drift correction between RM receptions. Specifically, a drift estimate $\overline{d}_{k,\text{RM}}$ at time instant k can be estimated from the time difference of successive RMs:

$$\overline{d}_{k,\text{RM}} = \frac{(\hat{N}_{k,\text{Si}} - \hat{N}_{k-1,\text{Si}}) \cdot T_{\text{Si}} - (\hat{t}_{k,\text{TM}} - \hat{t}_{k-1,\text{TM}})}{\hat{t}_{k,\text{TM}} - \hat{t}_{k-1,\text{TM}}}. \tag{5.6}$$

That is, (5.6) determines the accumulated time difference of Si with respect to TM, which is divided by the actual time passed between RMs. The number of HR ticks between LC updates P_{Si} can then directly be computed from $\overline{d}_{k,\mathrm{RM}}$ using (5.5). Here, we note that it is expected that P_{Si} slightly varies over time due to variations in the clock drift and inaccuracies in the timestamps.

5.4.2 Experimental Setup and Evaluation

We next evaluate the proposed AUTOSAR-DC method. The experimental setup consists of 3 MCU development boards [65] (nodes N1 to N3) and 1 FPGA development board (node N4) with an ARM Cortex-A9 processor [67]. Each board realizes a 32-bit LC with $T_N = 250\,\mathrm{ns}$, which is also chosen as the NTU.

The first experiment evaluates the clock differences Δc_N and drifts d_N of the CAN nodes without CS with respect to the LC of N1 as the TM over a time interval of 1 800 s. The remaining nodes N2, N3 and N4 are the slaves S1, S2 and S3. Fig. 5.10 shows that the clock drift of each slave is almost constant with the measured values $d_{\mathrm{S1}} \approx -35.2\,\mathrm{ppm}$, $d_{\mathrm{S2}} \approx -24.8\,\mathrm{ppm}$ and $d_{\mathrm{S3}} \approx -113.1\,\mathrm{ppm}$.

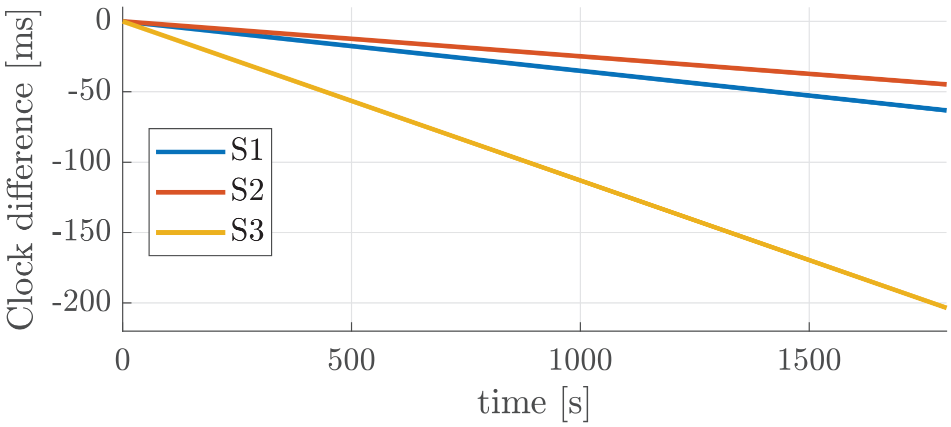

Figure 5.10: Clock drift measurement for the experimental setup.

We next illustrate the performance of the AUTOSAR method [27]. Fig. 5.11 shows a representative snapshot of the clock difference of the slave nodes for a RM period of $T = 1.048\,\mathrm{s}$, a bit rate of $B = 250\,\mathrm{kbps}$. It can be seen that the clock difference of each LC is reset to approximately zero with each RM in compliance with (5.3). Nevertheless, the LCs drift freely until the next RM such that a maximum (absolute) clock difference according to (5.2) (dashed line) occurs.

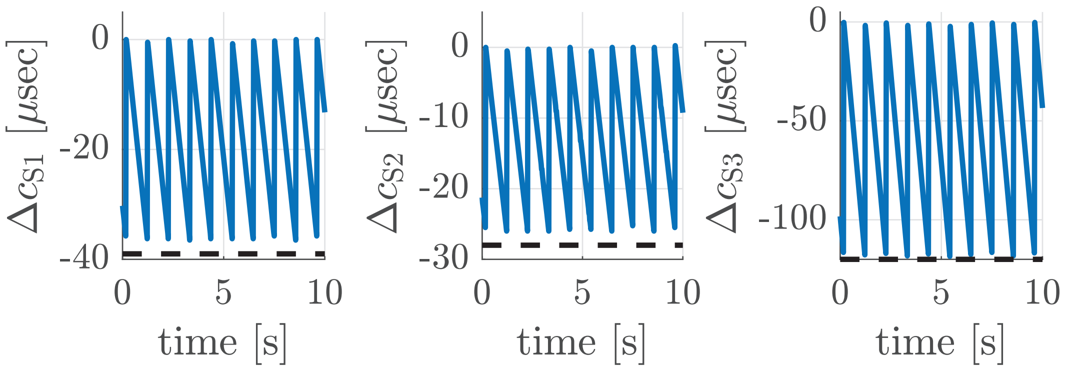

Figure 5.11: Clock difference for the AUTOSAR method.

We also evaluate the benefit of drift correction using AUTOSAR-DC in Section 5.4.1 on S3 for $\gamma = 1$. Fig. 5.12 shows clock differences and estimated clock drifts in (5.6) for different RM periods. The clock accuracy is significantly improved compared to the AUTOSAR method due to the applied drift correction as can be seen in the left and center plots. In particular, a clock accuracy in the order of $4\,\mu$s is achieved. Nevertheless, it is the case that the drift estimate is not precise and deteriorates for smaller RM periods (right plots) due to the increased effect of timestamping inaccuracies. In particular, decreasing the RM period does not necessarily improve the clock accuracy.

5.5 Controller-based Clock Synchronization for the Controller Area Network

In this section, a drift estimation approach based on feedback control is introduced. The main contribution of this section is new software-based clock synchronization (SW-CS) methods for CAN based on periodically transmitted RMs and discrete-time control formulations. In addition to the formulations, it is also explained how to determine suitable controller coefficients for the constructed models.

5.5.1 Basic Idea

The main idea of the controller-based CS method is based on the observation that the oscillator drift d_{Si} of any slave node only shows small and slow variations over time. That is, knowing or estimating d_{Si}, it is possible to compensate the clock drift

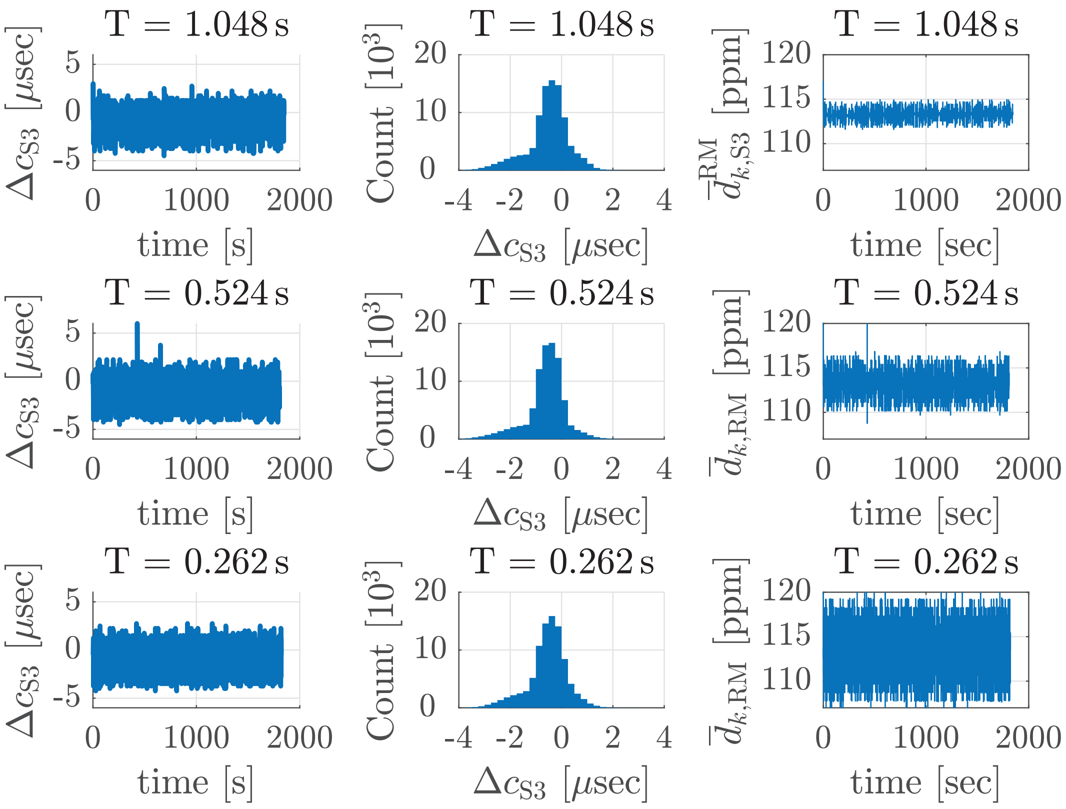

Figure 5.12: AUTOSAR-DC: Dependency on the RM period.

between RMs by continuously applying the drift correction $u(t) = -d_{\text{Si}}$ between RMs, different than the offset correction which applies clock corrections only after receiving RMs. Assuming that RMs are transmitted according to Gergeleit's method, we propose to compute the required drift correction $u(t)$ at the times t_k, $k = 0, 1, \ldots$ after the reception of a RM and to apply $u_k = u(t_k)$ for one cycle between t_k and t_{k+1}.

We formulate the computation as a discrete-time control problem for a generic slave node Si and the Master node TM in the feedback loop in Fig. 5.13. Here, $C(z)$ represents the drift correction controller and the gray box contains the LC model.

x_k in the LC model represents the clock difference at t_k as $c_{\text{Si}}(t_k) - c_{\text{TM}}(t_k)$ and the corresponding clock error is $e_k = -x_k$. Writing D_k for the overall clock drift of Si with respect to TM and $T \cdot u_k$ for the overall clock correction between t_k and t_{k+1} the discrete-time model of the clock difference in Si is given by

$$x_{k+1} = x_k + D_k + T \cdot u_k. \tag{5.7}$$

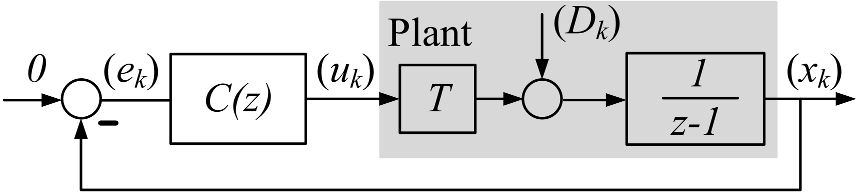

Figure 5.13: Feedback loop.

Applying the z-transform with the signals $U(z) \bullet\!\!-\!\!\circ\, u_k$ and $X(z) \bullet\!\!-\!\!\circ\, x_k$ in the z-domain, the plant transfer function is

$$G(z) = \frac{X(z)}{U(z)} = \frac{T}{z-1}. \tag{5.8}$$

We further note that the available information for computing u_k at time t_k consists of the previous drift correction values $u_{k-1}, u_{k-2}, u_{k-3}, \ldots$ and the clock error measurements $e_{k-1}, e_{k-2}, e_{k-3}, \ldots$. Hence, we define the controller as

$$u_k = c_1\, u_{k-1} + c_2\, u_{k-2} + c_3\, u_{k-3} + g_1\, e_{k-1} + g_2\, e_{k-2} + g_3\, e_{k-3}. \tag{5.9}$$

Here, c_1, c_2, c_3, g_1, g_2, g_3 are the unknown controller coefficients. We note that the measurement e_k is not available at t_k due to the application of Gergeleit's method for timestamping. The controller transfer function with $E(z) \bullet\!\!-\!\!\circ\, e_k$ is

$$C(z) = \frac{U(z)}{E(z)} = \frac{g_1\, z^2 + g_2\, z + g_3}{z^3 - c_1\, z^2 - c_2\, z - c_3} = \frac{P(z)}{L(z)}. \tag{5.10}$$

5.5.2 Discrete-time Control Problem

In order to determine suitable controller coefficients, we employ the disturbance sensitivity

$$\frac{X(z)}{D(z)} = \frac{z^3 - c_1\, z^2 - c_2\, z - c_3}{z^4 + a_3\, z^3 + a_2\, z^2 + a_1\, z + a_0} = \frac{L(z)}{A(z)} \tag{5.11}$$

with $a_0 = c_3 + T\, g_3$, $a_1 = c_2 - c_3 + T\, g_2$, $a_2 = c_1 - c_2 + T\, g_1$ and $a_3 = -1 - c_1$. There are two basic requirements for the choice of the controller parameters c_1, c_2, c_3, g_1, g_2, g_3.

1. The roots of the characteristic polynomial $A(z) = z^4 + a_3\, z^3 + a_2\, z^2 + a_1\, z + a_0$ need to be within the unit circle in order to achieve a stable feedback loop.

2. One root of the controller denominator $L(z)$ should be at $z = 1$ to obtain integrating behavior, which leads to a zero steady-state error for the given disturbance D_k.

In addition, a small magnitude of the roots of $A(z)$ is desired to achieve fast convergence of the clock error e_k to zero.

In this thesis, we apply optimization for finding suitable roots. First, we formulate requirement 2) as a constraint:

$$(z^3 - c_1\, z^2 - c_2\, z - c_3)|_{z=1} = 0 \Rightarrow c_3 + c_2 + c_1 = 1. \tag{5.12}$$

Writing $\mathcal{R}(A(z))$ for the set of roots of $A(z)$, we want to minimize the maximum absolute value of any root $r \in \mathcal{R}(A(z))$, while fulfilling (5.12). In order to adjust the convergence time we further employ the constraint in (5.15) such that the absolute value of any $r \in \mathcal{R}(A(z))$ stays above a minimum value r_{min}.

$$\min_{c_1, c_2, c_3, g_1, g_2, g_3} \{ \max_{r \in \mathcal{R}(A(z))} |r| \} \tag{5.13}$$

subject to

$$c_3 + c_2 + c_1 = 1 \tag{5.14}$$

$$|r| \geq r_{\text{min}}, \forall r \in \mathcal{R}(A(z)) \tag{5.15}$$

The controllers in this section are all computed by solving the nonlinear optimization problem in (5.13) to (5.15) for selected values of r_{min} using Matlab. We note that this problem is solved offline and the found parameters can then be implemented in software on any MCU.

5.5.3 Experimental Setup

We validate the controller-based CS method using the CAN network with the nodes N1, N2, N3 and N4. It consists of 3 MCU development boards [65] (N1-N3) and 1 FPGA development board (N4) with an ARM Cortex-A9 processor [67]. Each board realizes a LC in software with a width of 32 bit and a time tick of $T_{\text{N}} = 250$ ns while N1=TM, N2=S1, N3=S2, N4=S4. Since the MCU boards have a nominal oscillator frequency of $f_{\mathrm{CO,N}} = 4\,\mathrm{MHz}$, the number of CO ticks per HR increment is $n_{\mathrm{HR,N}} = 1$ for N1, N2, N3. The FPGA development board has an oscillator with $f_{\mathrm{CO,N4}} = 33.3\,\mathrm{MHz}$ such that $n_{\mathrm{HR,N4}} = 8.25$ is adjusted with the help of the Phase Locked Loop (PLL) modules.

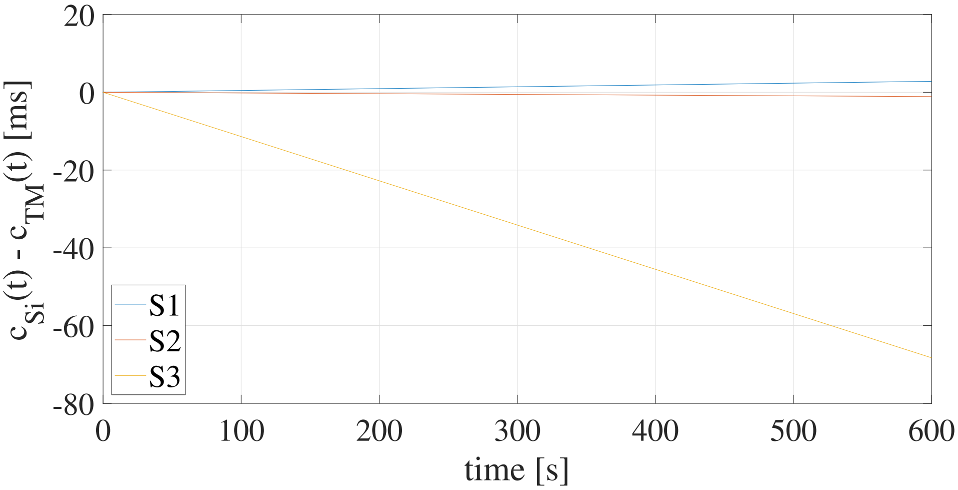

Figure 5.14: Clock drift.

Fig.5.14 shows the clock drift of the nodes S1,S2 and S3 in this experiment when a CS method is not followed. Accordingly, the clock drifts are measured as 4.7$\mu s/s$, -1.83$\mu s/s$ and -113.8$\mu s/s$ for S1, S2 and S3, respectively.

5.5.4 Basic Evaluation of the Existing CS methods

In order to evaluate and compare the proposed controller-based CS method, the clock accuracy performance of the existing fundamental CS methods are measured first on the constructed experimental setup. Accordingly, using the Gergeleit's CS method [28] and AUTOSAR CS method [27], slaves correct LCs at the periodic time instants t_k, $k = 0, 1, \ldots$. In between these time instants, the LC of each slave drifts because of the oscillator drift.

We illustrate this fact using the setup that is explained in 5.5.3. We choose TM = N1 as the perfect clock and implement the Gergeleit's CS method [28] and AUTOSAR CS method [27] on the slave nodes. Then, we measure the clock difference $x_{\mathrm{S3}}(t) = c_{\mathrm{S3}}(t) - c_{\mathrm{TM}}(t)$ for the slave S3 = N4 at a bit rate of $B = 250\,\mathrm{kbps}$ and a bus load of $U = 70\%$ generated by all the nodes. Fig. 5.15 shows representative results for $T = 1.048\,\mathrm{s}$, $T = 0.524\,\mathrm{s}$ and $T = 0.262\,\mathrm{s}$.

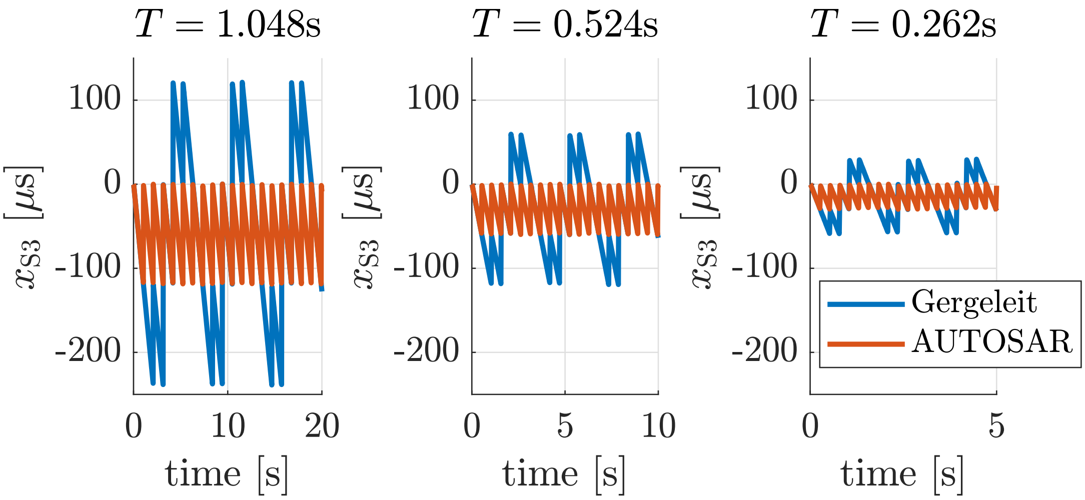

Figure 5.15: Comparison of existing methods.

Considering that the clock drift for S3 is $d_{\mathrm{S3}} \approx -113.8\,\mathrm{ppm}$ as measured in Section 5.5.3, maximum clock differences in the order of $2 \cdot T \cdot d_{\mathrm{S3}}$ and $T \cdot d_{\mathrm{S3}}$ are observed for Gergeleit's method and the AUTOSAR method, respectively. The AUTOSAR method takes timestamps very close to the update time such that $c_{\mathrm{S3}}(t_k^+) \approx c_{\mathrm{TM}}(t_k^+)$ right after t_k. Then, the LC of S3 drifts with d_{S3} for the time T. Differently, the timestamp in Gergeleit's method is taken one cycle before t_k. That is, generally $c_{\mathrm{S3}}(t_k^+) \not\approx c_{\mathrm{TM}}(t_k^+)$, which causes the increased clock difference. The advantage of Gergeleit's method is the requirement of a single RM per cycle.

A common shortcoming of all the RM-based offset correction methods is that they only correct the LC right after the reception of a RM, whereas the clocks drift freely between RMs.

5.5.5 Implementation Details and Evaluation

Any slave node that realizes the proposed controller-based CS method has to both perform timestamping as in Gergeleit's method and continuously apply the drift correction in (5.9). We suggest to use corrections in integer multiples γ of the time resolution T_{N} of the LC. That is, making corrections of $\gamma \cdot T_{\mathrm{N}}$, each correction has to be carried at time intervals of

$$\frac{\gamma \cdot T_{\mathrm{N}}}{u_k}. \tag{5.16}$$

Hereby, γ can be chosen respecting the trade-off between precision (frequent corrections) and processor load (infrequent corrections). Considering that $T_{\mathrm{N}} = 250\,\mathrm{ns}$ in our implementation, we choose $\gamma = 4$. That is, for a drift correction of 113.8μs/s, the update time is $8.8\,\mathrm{ms}$ (note that the update time dynamically changes in every cycle k depending on u_k).

Using the described implementation, we first perform an experiment with different RM periods $T = 1.048\,\mathrm{s}$, $T = 0.524\,\mathrm{s}$ and $T = 0.262\,\mathrm{s}$. In this experiment, a bus load of $U = 70\,\%$ is generated by the nodes and the bit rate is $B = 250\,\mathrm{kbps}$. The controller parameters were computed for $r_{\text{min}} = 0.5$ in (5.13) to (5.15). Fig. 5.16 shows a comparison of simulations in Matlab/Simulink with the clock difference measurements from the CAN node S3 in the experimental setup.

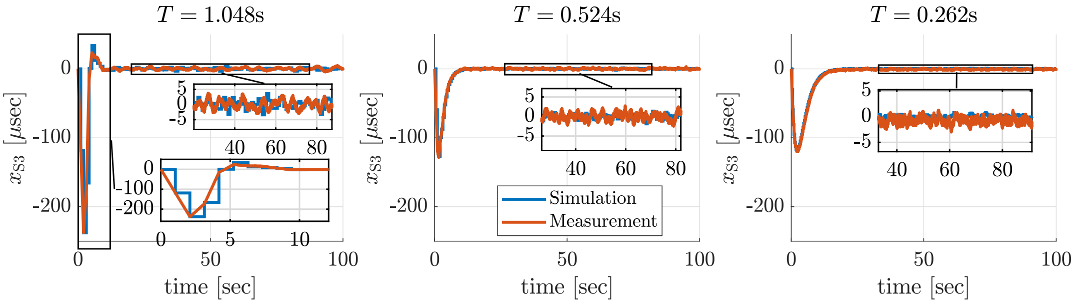

Figure 5.16: Comparison of simulation and measurements for different values of T for $B = 250\,\mathrm{kbps}$ and $U = 70\%$.

It can be seen that the maximum clock difference is well below $5\,\mu$s for all values of T when reaching the steady state after starting up the nodes. Considering that $d_{\mathrm{S3}} \approx -113.8\,\mathrm{ppm}$ for S3, this is a major improvement compared to the existing methods, which cannot achieve maximum clock differences below $|d_{\mathrm{S3}}| \cdot T$ (see Fig. 5.15). The main reason for this improvement is the ability of correcting the clock drift between RMs. Nevertheless, there is still one shortcoming of the proposed algorithm at the start-up of slave nodes. It takes about $10\,\mathrm{s}$ until the clock difference converges to the steady state. We address this shortcoming in the next section.

5.5.6 Initialization Phase

The slow convergence to the steady-state value is mostly due to fact that the values of the signal values $u_{k-3}, u_{k-2}, u_{k-1}, e_{k-3}, e_{k-2}, e_{k-1}$ required by the controller are undefined at start-up. We next develop a heuristic procedure in order to determine suitable values within at most 3 RM periods as shown in Algorithm 5.

Algorithm 5 Drift Correction (DC) with Initialization

```
1:  Input: $e_{k-1}, e_{k-2}, e_{k-3}, T$
2:  Output: $u_k$
3:  Init: $u_{k-1} = 0$; $u_{k-2} = 0$; $u_{k-3} = 0$; $u_k = 0$; $k = 0$
4:  if RM is received then
5:      $k = k + 1$; $e_{k-1} = t_{k-1,\text{TM}} - t_{k-1,\text{Si}}$
6:      if $k = 3$ (3$^{\text{rd}}$ RM is received at $t_3$) then
7:          $u_k = (e_{k-1} - e_{k-2})/T$; $\hat{e}_k = e_{k-2} + (2 \cdot T \cdot u_k)$
8:          $c_{\text{Si}}(t_k) = c_{\text{Si}}(t_k) + \hat{e}_k$ (clock update)
9:          $\hat{t}_{k,\text{Si}} = \hat{t}_{k,\text{Si}} + \hat{e}_k$ (timestamp overwrite)
10:     else if $k > 3$ then
11:         Calculate $u_k$ according to (5.9)
12:     end if
13: end if
```

Algorithm 5 records the error values starting from the first RM received ($k = 1$). No drift correction is applied until the third RM (line 3 and 5). With the third RM (line 6), the algorithm estimates the input signal u_k and the clock error $\hat{e}_k$ at t_k from the difference of the previous clock errors e_{k-1} and e_{k-2} (line 7). The LC and timestamp at t_3 are then updated using $\hat{e}_k$. After t_3, the proposed drift correction in (5.9) is applied (line 11). A main advantage of the proposed initialization procedure is that a small clock difference is achieved after receiving 3 RMs when starting up a CAN node.

5.5.7 Further Evaluation with Initialization Phase

We next perform a comprehensive evaluation of the proposed control method. We first study the convergence to the steady state with and without the initialization procedure. The clock difference measurements for node S3 and RM periods $T = 1.048\,\mathrm{s}$, $T = 0.524\,\mathrm{s}$ and $T = 0.262\,\mathrm{s}$ are shown in Fig. 5.17. Different from the original control method, the steady state is reached within at most $3\,T$ for the algorithm with initialization. In particular, choosing smaller values of T allows decreasing the start-up time without any negative effect on the clock differences in the steady state, which are well below $5\,\mu\mathrm{s}$. We note that using line 7 to 9 in Algorithm 5 also in the steady state leads to a decreased CS performance.

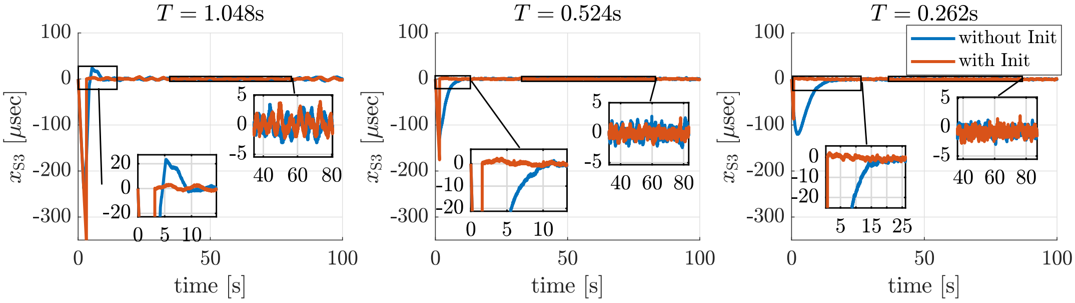

Figure 5.17: Comparison of the control algorithm with and without initialization for $B = 250\,\mathrm{kbps}$ and $U = 70\%$.

For further analysis, the upper part of Fig. 5.18 compares the histograms of the clock differences for the AUTOSAR (ASAR) method and our controller-based CS algorithm (DC) with initialization for different RM periods with $U = 70\%$ and $B = 250$ kbps.

While the clock differences for the AUTOSAR method are almost equally distributed with maximum values around $T \cdot d_{\mathrm{S3}}$, our algorithm leads to clock differences that are tightly centered around 0. Consider for example the AUTOSAR method for $T = 0.262\,\mathrm{s}$ and the proposed method for $T = 0.524\,\mathrm{s}$. Even two messages (SYNC and RM) are transmitted every $0.262\,\mathrm{s}$ using the AUTOSAR method, the maximum clock difference of about $31.25\,\mu\mathrm{s}$ is observed periodically before each RM. Differently, only one RM is sent every $0.524\,\mathrm{s}$ using our method, whereas the clock difference stays below $2.5\,\mu\mathrm{s}$ after the initialization period of $3\,T = 1.572\,\mathrm{s}$. This is a major

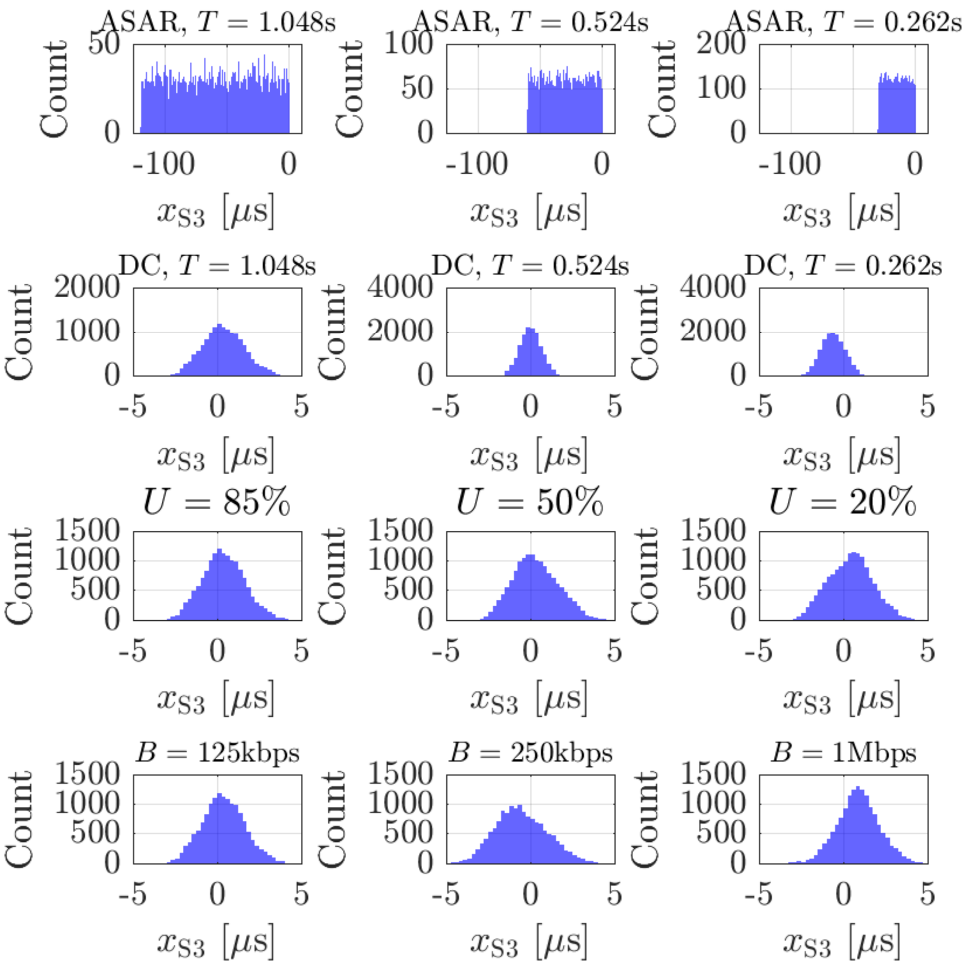

Figure 5.18: Clock differences for different parameters.

performance boost (factor of 50) of SW-CS for CAN. We note that clock differences in the order of 5μs are reported for existing methods [63]. However, this is only possible for very small RM periods and CAN nodes with very similar COs. This is not realistic in practical CAN networks with CAN nodes from diverse manufacturers. Conversely, our method is independent of the clock drift of COs.

We finally investigate the dependency of the proposed controller-based CS method on the bus load (for $T = 1.048$s and $B = 250$ kbps) and the bit rate (for $T = 1.048$s and $U = 70\%$) as shown in the lower part of Fig.5.18. It is clearly observed that there is no significant dependency of the clock difference on the bus load and the bit rate. In all cases, the clock difference stays below $5\,\mu$s, which is one order of magnitude better than the clock difference obtained using the AUTOSAR method.

In this section, a clock synchronization method based on feedback control is presented. It proposes estimating the clock drift with a controller that benefits from the timestamps which are periodically taken together with RMs. To this end, we formulate a discrete-time control problem and develop a start-up procedure for fast convergence to the steady-state. Different from existing algorithms that only update clocks after receiving a RM, our algorithm corrects the clock drift between RMs.

5.5.8 AUTOSAR CS Compatible Controller Model and Parameters

The controller model that is proposed for the timestamps compatible to the Gergeleit's scheme can be modified such that it works with timestamps that are sent according to the AUTOSAR CS method. That is, for computing u_k at time t_k the previous drift correction values $u_{k-1}, u_{k-2}, ...$ and the clock error measurements $e_k, e_{k-1}, ...$ are reachable. It has to be noted that e_k measurement which is not usable when the timestamps are applied according to the Gergeleit's method is available when the timestamps are taken and transmitted as defined by the AUTOSAR CS method. Thus, the controller can be defined as

$$u_k = c_1\, u_{k-1} + g_1\, e_k + g_2\, e_{k-1}. \tag{5.17}$$

We employ the disturbance sensitivity by following again the same model in Fig.5.13 as follows:

$$\frac{X(z)}{D(z)} = \frac{z - c_1}{z^2 + z\,(T\,g_1 - c_1 - 1) + c_1 + T\,g_2} = \frac{L(z)}{A(z)} \tag{5.18}$$

The roots of the characteristic polynomial $A(z)$ should be within the unit circle to achieve a stable feedback loop. Moreover, one root of $L(z)$ should be at $z = 1$ which provides a zero steady state error for the given disturbance D_k. Accordingly, the suitable controller parameters are computed offline such that $c_1 = 1$, $g_1 = 1.24$ and $g_2 = -0.84$ for $T = 1.048\,\mathrm{s}$.

Furthermore, the initialization procedure in Algorithm 6 should be used. Different from the initialization procedure for the Gergeleit's scheme, the procedure 6 predicts the input signal u_k after receiving the second FUP message. Furthermore, it is not required to estimate the clock error $\hat{e}_k$ at t_k since it is already available when AUTOSAR CS is applied.

Algorithm 6 Drift Correction (DC) with Initialization - AUTOSAR compatible

```
Input: $e_k, e_{k-1}, T$
Output: $u_k$
Init: $u_{k-1} = 0; u_k = 0; k = 0$
if RM is received then
    $k = k + 1; e_k = t_{k,\text{TM}} - t_{k,\text{Si}}$
    if $k = 2$ ($2^{\text{nd}}$ FUP is received at $t_2$) then
        $u_k = (e_k - e_{k-1})/T$
        $c_{\text{Si}}(t_k) = c_{\text{Si}}(t_k) + e_k$ (offset correction)
        $\hat{t}_{k,\text{Si}} = \hat{t}_{k,\text{Si}} + e_k$ (timestamp overwrite)
    else if $k > 2$ then
        Calculate $u_k$ according to (5.17)
    end if
end if
```

The developed AUTOSAR compatible controller-based method is also evaluated with comprehensive experiments. It is seen that the clock accuracy is achieved below $4\,\mu$s independently from the RM period, bus load and bus speed like the Gergeleit's method compatible controller-based CS. The local clock difference of S3 is shown in Fig.5.19 as an illustration where $T = 1.048\,\text{s}$, $B = 250\,\text{kbps}$ and $U = 70\%$.

5.6 Dependency of the CS on Timestamping Quality

It can be concluded from the previous discussion that CS methods on CAN depend on accurate timestamps taken by all CAN nodes, whereby it is expected that the quality of the timestamps directly affects the clock accuracy. This effect is well known for different networks such as IEEE 802.11 [105, 106] but has not been rigorously evaluated for CAN. The existing literature only suggests to use hardware TS [58, 43] and to benefit from jitter measurements in case of software TS [19] without providing an

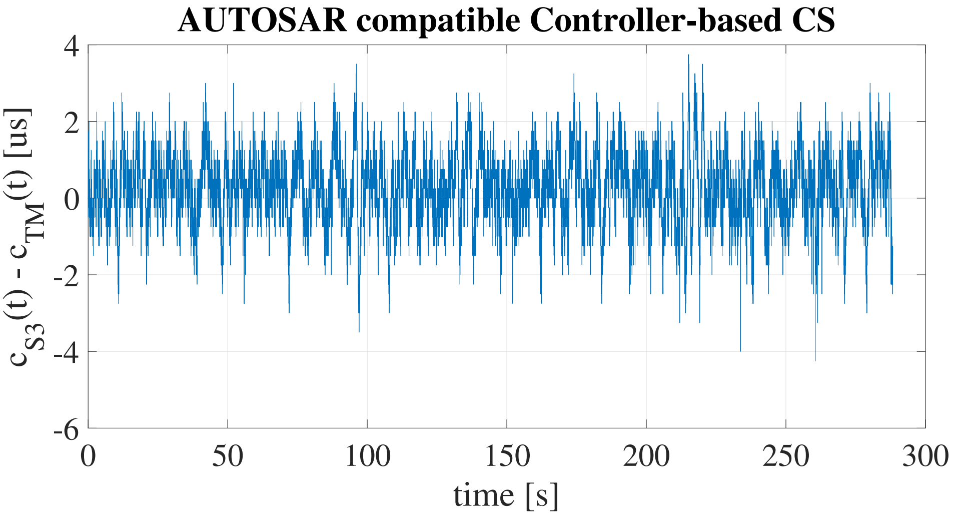

Figure 5.19: The clock difference results.

analysis. Accordingly, this section shows that the clock accuracy is directly affected by the TS quality, which in turn depends on the uncertainties of each TS method. Moreover, our predictable TS method which is introduced in Chapter 4 makes possible for our controller-based CS method in Section 5.5.8 to achieve clock accuracies below 100 ns.

5.6.1 Experimental Evaluation

The experimental setup described in 4.8.1 is used to evaluate the effect of the TS quality on the CS accuracy. The controller-based clock synchronization method in Section 5.5.8 is used in the experiments. The RMs are sent according to the Autosar CS method [27] and the controller parameters are computed accordingly. A 32-bit LC with a resolution of 100 ns is used in the experiments. A practical value of 1 s is chosen as the RM period in the experiments and the clock accuracy is monitored with a period of 20 ms. We note that the oscillator drifts of the slave nodes S_1, S_2 and S_3 relative to TM are measured as 98, 99 and 96 ppm, respectively. That is, a clock drift in the order of 100 μs per second is expected without CS.

Firstly, an experiment where the CAN bus speed is 1 $Mbps$ and TQ length of all nodes is 50 ns is realized with our predictable TR method and hardware TS service implementation. The TS quality in the order of 60 ns is measured . However, the experiment is repeated with a change that TS service is implemented on software. In this case, the TS quality decreases to the order of 900 ns due to the varying software delays. The effect of the TS quality on the CS accuracy can be seen from the clock accuracy results in Table 5.1. As a consequence, the controller-based CS method achieves a peak-to-peak clock accuracy of 120 ns with higher TS quality (hardware implementation - TS quality of 60 ns). Nevertheless, the same CS method cannot provide a peak-to-peak clock accuracy better than 1.35 μs when the TS service is realized in software (TS quality of 900 ns). That is, the timestamp quality is one of the key parameters that determines the performance of the CS method.

Table 5.1: Clock accuracy results.

	Predictable TR + Hardware TS service (TS quality of 60 ns)			Predictable TR + Software TS service (TS quality of 900 ns)		
	S1	S2	S3	S1	S2	S3
mean(ns)	20.72	17.76	20.60	100.81	104.19	106.55
std.(ns)	16.94	17.43	16.73	182.07	186.69	187.07
max(ns)	70.00	80.00	80.00	765.63	750.00	765.63
min(ns)	-50.00	-40.00	-40.00	-718.75	-609.38	-609.38
p-p(ns)	120.00	120.00	120.00	1484.38	1359.38	1375.01

Moreover, the clock accuracy statistics of the experiments whose TS quality results are already presented in Table 4.2 are given in Table 5.2. It can be seen that the best clock accuracy results are obtained when using our predictable TS method. Furthermore, the correlation between the TR instant quality measurements and the clock accuracy results confirms the dependency of the CS performance on the TS quality. As it is expected, the CS accuracy gets worse for the lower bit rates since the TS quality is lower due to the increase in SYU uncertainties with higher TQ lengths, as explained in Chapter 4.

As an illustration, we further present the clock difference measurements in Fig.5.20 when applying the controller-based CS method with the predictable TR method with a bus speed of 1 Mbps. The clock difference for all slave nodes is mostly between 50

Table 5.2: Clock accuracy - overall comparison.

B		Node	EoM	SoF	Predictable TR
1Mbps	p-p(ns)	S1	140	120	120
		S2	140	120	120
		S3	170	270	120
	std(ns)	S1	21,9723	16,8967	16,9446
		S2	20,4889	17,0974	17,7649
		S3	29,5527	20,5308	20,6048
500kbps	p-p(ns)	S1	280	240	190
		S2	200	240	180
		S3	320	240	180
	std(ns)	S1	45,329	30,0388	29,7535
		S2	45,0746	30,7466	31,3127
		S3	59,4221	32,2348	30,9108
250kbps	p-p(ns)	S1	400	390	320
		S2	400	380	370
		S3	550	490	340
	std(ns)	S1	65,0135	60,1449	51,0321
		S2	69,0876	58,0711	67,6625
		S3	90,7366	61,0517	67,6835
125kbps	p-p(ns)	S1	1080	940	540
		S2	990	940	570
		S3	1050	610	550
	std(ns)	S1	165,3906	144,8683	105,0069
		S2	169,477	119,6268	110,1489
		S3	187,4457	135,9121	104,3040

ns and 0 ns in Fig.5.20 even though the peak-to-peak difference is 120 ns according to Table 5.1. To sum up, this constitutes the best clock accuracy performance for CAN in the literature under realistic conditions such as additional cables with the help of our new predictable TR instant method.

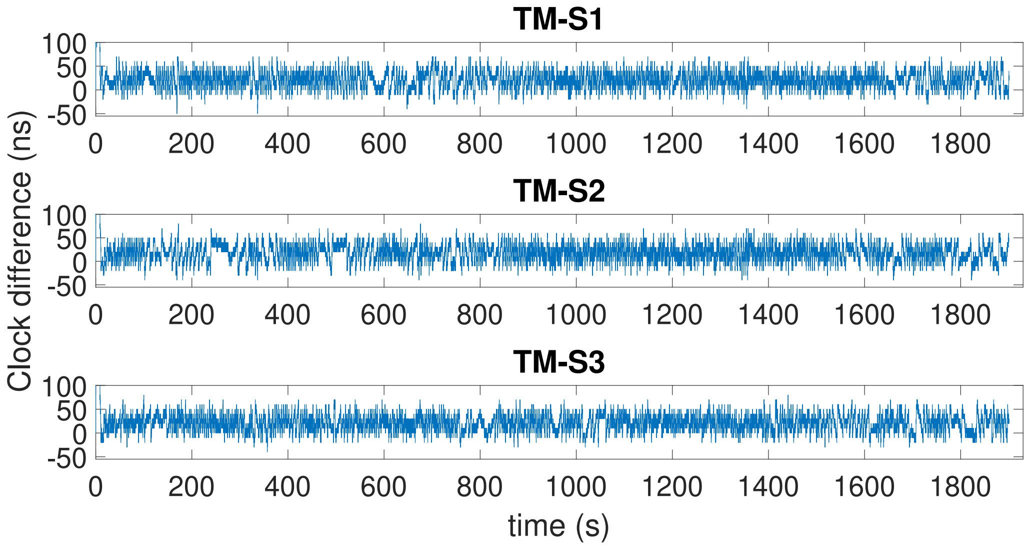

Figure 5.20: The clock difference results.

5.7 Discussion and Conclusion

In summary, this section provides several CS methods that are fully implemented in software. Additional to the leading offset correction CS methods AUTOSAR and Gergeleit's method, our ISCS method is proposed. Furthermore, it is experimentally shown that our ISCS provides the same clock accuracy with the AUTOSAR CS method whereas ISCS requires half of the RM bandwidth usage of the AUTOSAR. Moreover, ISCS performs better than Gergeleit's method even though they both use only one RM in one cycle.

The offset correction methods such as ISCS, AUTOSAR CS and Gergeleit's method cannot avoid clock drift between RMs. Thus, their performances are determined by the physical oscillator drifts of the nodes. In order to reach better clock accuracy independently from the oscillator drift values by applying drift correction, the RM-based CS method that realizes both offset correction and drift correction is introduced.

It is offered according to the timestamps that are exchanged as compatible to the AUTOSAR CS scheme. It estimates the drift value by simply using the rate of the changes in the timestamps. Moreover, the clock difference of a generic slave node relative to a master node is also modeled as a discrete time control problem. Both controller parameters and initialization procedures that are compatible to AUTOSAR CS and Gergeleit's method timestamp schemes are also developed. The correctness and practicability of the controller-based drift correction methods are shown with comprehensive experiments.

Lastly, the importance of the TS quality for CS performance is presented experimentally. Thus, TS quality is highly relevant for the SW-CS methods based on timestamps on CAN. The proposed SW-CS methods which mitigate the drift between RMs provide generally clock accuracies less than $5\,\mu$s with moderate quality timestamps. On the other hand, high quality timestamps which are achieved with our predictable TR method enable even clock accuracies below $100\,$ns. In brief, the proposed CS advancements are significant for CANDS framework that aims to apply traffic shaping with a precise global clock to realize a deterministic CAN bus.

CHAPTER 6

CLOCK SYNCHRONIZATION FOR THE CONTROLLER AREA NETWORK USING BIT TIMING INFORMATION

The main contribution of this section is a novel CS method for CAN. The proposed method performs both offset and drift correction. While offset correction is performed based on timestamps in periodic RMs, our new ACS-PEDC method benefits from the re-synchronization mechanism of the CAN bit timing to apply highly accurate drift correction. Therefore, the drift correction is not subject to TS inaccuracies as an advantage, different from the existing CS methods on CAN. Furthermore, our algorithm does not make any modifications to the CAN protocol but requires the measurement of the phase error from the CAN controller. We note that existing commercial CAN controllers do not provide the phase error information. However, since the phase error is computed in each CAN controller, only a minor modification of existing CAN controllers is required in order to make the phase error available.

Existing drift correction methods for CAN compute drift estimates only from timestamps with the following disadvantages. First, the accuracy of drift estimates directly depends on the timestamp accuracy, which can be in the order of μs even in hardware implementations [43]. Second, the drift estimate can only be updated infrequently after receiving a new RM. To mitigate these disadvantages, ACS-PEDC method with highly accurate and timely drift estimation is introduced. Specifically, our original drift estimation algorithm measures the time intervals between re-synchronizations of the CAN bit time and computes highly accurate drift estimates by the real-time computation of the approximate greatest common divisor (AGCD) of these time intervals. We further extend our algorithm by a procedure for the quick start-up in practical applications and derive analytical bounds for the expected clock accuracies.

Our comprehensive hardware experiments validate that our method achieves a clock accuracy below $2\,\mu$s after receiving only 2 RMs, independent of the bit rate, RM period, bus load, oscillator drift and changes in the ambient temperature. As a major advantage, our method does not require modifications of the CAN protocol such that CAN nodes implementing our algorithm can operate on a standard CAN network. To sum up, the proposed novel drift estimate method enables CANDS to predict the clock drift precisely without the need of timestamps within RMs. In addition to the proposed SW-CS methods which are previously evaluated in Chapter 5, the new original ACS-PEDC method that benefits from the bit timing information can also be followed within CANDS without modifying the CAN protocol in order to provide a global clock among the nodes.

6.1 Basic Idea behind the Phase-error Based Drift Correction

The RM-based drift correction method in Section 5.4.1 requires a correction period P_{Si} and the sign of the drift s_{Si} in order to apply drift correction. Hereby, P_{Si} is represented in terms of HR ticks of Si's LC, which is used for timestamping. It has to be remembered that one HR tick clock difference is accumulated between LCs of Si and TM after LC of Si increments P_{Si} HR ticks. Nevertheless, the same idea can be applied using the TQ counter of the CAN system clock which is described in Section 2.1.3. Specifically, a difference of s_{Si} TQ counter ticks is accumulated during the time $s_{\mathrm{Si}} \cdot T_{\mathrm{TQ,Si}}/d_{\mathrm{Si}}$ according to the clock drift equation 2.5. Hereby, P_{Si} TQ counter ticks in Si equal to the time duration of $s_{\mathrm{Si}} \cdot T_{\mathrm{TQ,Si}}/d_{\mathrm{Si}}$. Therefore, the time duration during which one TQ counter tick accumulates between Si and TM directly gives the clock drift of Si.

In this context, the crucial observation in this chapter is that, the amount of the clock drift of the TQ counter in Si relative to TM can be determined by using the phase error values e_{P} while re-synchronizations during message transmission of the TM. Furthermore, the clock drift of LC of Si relative to TM is found when LC and TQ counters are driven by the same oscillators on Si and TM nodes.

We denote the time instant when TQ counters of both Si and TM are aligned as phase error overflow (PEO) since the previous e_{P} values equal to 0 but the upcoming e_{P} value will be different than 0 due to the drift of TQ counters in Si and TM. Thus, a clock difference of s_{Si} TQ ticks is accumulated in LC of Si relative to LC of TM between two successive PEO instants. Furthermore, the sign of e_{P} right after the PEO instant gives the sign of the clock drift Si s_{Si}. In this way, the clock drift correction period P_{Si} in TQ counter ticks can be determined by benefiting from the internal bit timing mechanism on CAN.

The PEO time instants can be detected by a slave node Si by trying to catch the time instants where e_{P} changes from 0 to ± 1, during message transmission of the TM . Accordingly, Si can determine P_{Si} as the number of TQ ticks between PEO instants. That is, P_{Si} does not need to be computed from the periodic timestamps but can be directly measured using the PEO instants.

Nevertheless, it has to be respected that Si can only detect PEO when applying resynchronization during message transmissions of the TM since the TQ counter of Si is synchronized to the node which is transmitting on the bus. TM does not transmit messages continuously and it is possible that Si misses PEOs that occur while other nodes are transmitting or while the bus is idle. Writing $\widehat{\mathrm{PEO}}$ for the detected PEO instants, we introduce Δq_j, $j = 1, 2, \ldots$ as the number of TQs of Si between successive $\widehat{\mathrm{PEO}}$s. Considering the possibility of missing PEOs, the measurements Δq_j are hence expected to be approximate multiples of P_{Si}.

Remark 7 *The concept of PEO requires that the length of TQs of TM is a multiple of the length of TQs of Si to ensure that the time between successive PEOs is determined only by the clock drift of Si. Otherwise, it may also be affected due to the bit-level synchronization mechanisms during message transmissions of other nodes. This is not a restriction since even simple CAN controllers allow adjusting the TQ length.*

We next consider the computation of estimates $\overline{P}_{\mathrm{Si}}$ of P_{Si} after each measurement Δq_j. When successive PEOs (without missing any PEO) are detected by Si, the related measurement Δq_j is almost equal to P_{Si}. Thus, a naïve approach for the estimation of P_{Si} would be to record the smallest observed values Δq_j to compute

$$P_{\mathrm{Si}} \approx \overline{P}_{\mathrm{Si}} = \min_{j=1,2,\ldots} \{\Delta q_j\}. \tag{6.1}$$

Nevertheless, such approach is neither efficient nor robust to uncertainties in practice.[1] On the one hand, there is a potentially long delay until successive PEOs are detected by Si. On the other hand, only the smallest values of Δq_j are used for the drift estimation and multiples of P_{Si} are discarded by the naïve approach although they contain information about P_{Si}.

6.2 Usage of the Approximate GCD

In order to mitigate the discussed shortcomings of the naïve approach, we next develop a practical algorithm that makes use of all the measurements Δq_j and the related values of the phase error e_{P}. Our algorithm takes into account that Δq_j must be an approximate multiple of P_{Si}. Formally, $\Delta q_j = n_j \cdot P_{\mathrm{Si}} + \Delta p_j$, where Δp_j represents a bounded measurement uncertainty.

We next quantify Δp_j. Consider that the TM transmits a message. Then, it holds that PEO for an arbitrary slave Si can occur during any bit of the message. However, PEO can only be detected when Si performs re-synchronization with the first falling edge after the PEO. The worst-case scenario is shown in Fig. 6.1.

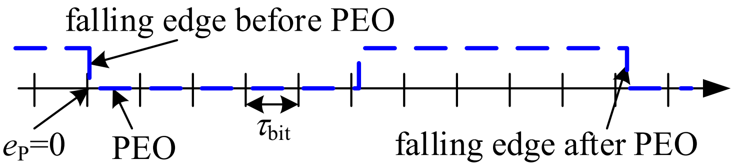

Figure 6.1: Illustration of the PEO detection.

Here, PEO occurs right after a falling edge such that it can only be detected with the next falling edge on the CAN bus. Hereby, the duration until the next falling edge is limited due to the bit stuffing mechanism on CAN. Specifically, a bit of the opposite value is inserted after five consecutive bits with the same value for all fields of the CAN frame except for the CRC delimiter, ACK field and the EOF [4]. That is, in the worst case, the falling edge before PEO is followed by 5 0-bits and 5 1-bits before the next falling edge must occur due to bit stuffing. Recalling that Δp_j is counted in

---

[1] This is also demonstrated in Section 6.5.2.

TQs, it holds that

$$|\Delta p_j| \leq X := 10 \cdot \#_{\mathrm{TQ,Si}}. \tag{6.2}$$

In addition, it is possible to determine the smallest possible value of P_{Si} depending on the maximum tolerable clock drift $d_{\max}$, which is in the order of ±100 ppm [107, 108, 109] for clock oscillators in practical applications. Using (5.5), we obtain

$$\begin{aligned} P_{\mathrm{Si}} &= \left\lceil s_{\mathrm{Si}} + \frac{s_{\mathrm{Si}}}{d_{\mathrm{Si}}} \right\rceil \geq \left| \frac{1}{d_{\mathrm{Si}}} \right| - 1 \geq \\ &\geq Y := \frac{1}{d_{\max}} - 1 = \frac{1}{200 \cdot 10^{-6}} - 1 = 4\,999. \end{aligned} \tag{6.3}$$

Using the bounds X and Y, we next develop the novel idea of computing the AGCD [110] of the measurements Δq_j, $i = 1, 2, \ldots$ to determine P_{Si} by iteratively refined estimates $\overline{P}_{\mathrm{Si},j}$ that are also approximate multiples of P_{Si}. In each iteration, we want to obtain the AGCD $\overline{P}_{\mathrm{Si},j}$ of Δq_j and the previous estimate $\overline{P}_{\mathrm{Si},j-1}$ by finding unique coprime natural numbers $\overline{n}_j, \overline{m}_j$ such that $\frac{\Delta q_j}{\overline{n}_j} \approx \frac{\overline{P}_{\mathrm{Si},j-1}}{\overline{m}_j} \approx \overline{P}_{\mathrm{Si},j}$. To this end, the new result in Theorem 2 states sufficient conditions for a tight approximation of the greatest common divisor (GCD) of two natural numbers. We write $\mathbb{N}$ for the set of natural numbers (excluding zero), $\mathbb{Z}$ for the set of integers and $\lfloor \bullet \rceil$ for the nearest integer function.

Theorem 2 *Consider two natural numbers $N, M \in \mathbb{N}$ such that $N = n \cdot g + \Delta n$ and $M = m \cdot g + \Delta m$ for $m, n, g \in \mathbb{N}$, m, n coprime and $\Delta n, \Delta m \in \mathbb{Z}$. Further assume that there are bounds $X, Y \in \mathbb{N}$ such that $|\Delta n|, |\Delta m| < X$ and $g > Y$. Then, it holds that m, n are the unique coprime natural numbers such that*

$$|N \cdot m - M \cdot n| < X \cdot (m + n). \tag{6.4}$$

and

$$2 \cdot X \cdot (m + n) < Y. \tag{6.5}$$

In addition consider any estimate $\hat{g} = \left\lfloor \lambda \cdot \frac{N}{n} + (1 - \lambda) \cdot \frac{M}{m} \right\rceil$ of g for $0 \leq \lambda \leq 1$. Then,

$$|g - \hat{g}| < \left\lfloor \frac{X}{\min\{n, m\}} \right\rceil. \tag{6.6}$$

The proof of Theorem 2 is given in Appendix B.1. In words, Theorem 2 considers two numbers N, M that are approximate multiples of g. That is, $\frac{N}{n} \approx \frac{M}{m} \approx g$. Then, the theorem states that the multipliers n and m can be uniquely determined if a lower bound Y of g and an upper bound X for $\Delta n, \Delta m$ are known and the error bounds in (6.4) and (6.5) are fulfilled and n, m are coprime natural numbers. In addition, any weighted average $\hat{g}$ of $\frac{N}{n}$ and $\frac{M}{m}$ tightly approximates the GCD g of N and M. In our setting, g represents P_{Si}, N represents a measurement Δq_j, M represents a previous estimate $\overline{P}_{\text{Si},j-1}$ and X, Y are given in (6.2) and (6.3).

6.3 Approximate GCD-based Drift Correction

Using Theorem 2 and the bounds in (6.3) and (6.2), it is possible to obtain a tight and reliable estimate of P_{Si} based on the AGCD of the measurements Δq_j. The main idea of our method is to keep a certain number of recent measurements in a table v. At each iteration j, row r of v stores the values $\Delta q_k / r$ for previous measurements ($k \leq j$) that are confirmed as a multiple of the current estimate $\overline{P}_{\text{Si},j}$ with $\Delta q_k \approx r \cdot \overline{P}_{\text{Si},j}$ using Theorem 2. The current estimate $\overline{P}_{\text{Si},j}$ is then the average of the stored values.

As an explanatory example, consider the case where $X = 160$, $Y = 4\,999$, the table v has 5 rows and 2 columns and the successive PEO measurements $\Delta q_1 = 53\,040$, $\Delta q_2 = 17\,592$, $\Delta q_3 = 44\,532$, $\Delta q_4 = 17\,605$, $\Delta q_5 = 8\,763$, $\Delta q_6 = 26\,502$, $\Delta q_7 = 35\,330$, $\Delta q_8 = 26\,643, \ldots$ are obtained.

As the first estimate $\overline{P}_{\text{Si},1} = \Delta q_1 = 53\,040$, we directly use the first measurement and insert the corresponding value in row 1, entry $v[1,1]$ of the table in Fig. 6.2. Using (6.4) and (6.5) for the second measurement Δq_2, we find $\overline{m}_2 = 3$ and $\overline{n}_2 = 1$ such that $|\Delta q_2 \cdot \overline{m}_2 - \overline{P}_{\text{Si},1} \cdot \overline{n}_2| = 264 < X \cdot (\overline{m}_2 + \overline{n}_2) = 640$ and $2 \cdot X \cdot (\overline{m}_2 + \overline{n}_2) = 1\,280 < Y$. Since $\overline{m}_2 = 3$, we know that Δq_1 is 3 times the current estimate. It is hence moved to row 3, entry $v[3,1]$ in the table and its value is divided by 3. Since $\overline{n}_2 = 1$, Δq_2 is inserted in row 1, entry $v[1,1]$ of the table. The current estimate becomes $\overline{P}_{\text{Si},2} = 0.5 \cdot (v[1,1] + v[3,1]) = 17\,636$. For Δq_3, we find $\overline{m}_3 = 2$ and $\overline{n}_3 = 5$ to determine $|\Delta q_3 \cdot \overline{m}_3 - \overline{P}_{\text{Si},2} \cdot \overline{n}_3| = 884 < X \cdot (\overline{m}_3 + \overline{n}_3) = 1\,120$ and $2 \cdot X \cdot (\overline{m}_3 + \overline{n}_3) = 2\,240 < Y$. Since $\overline{m}_3 = 2$, the previous estimate $\overline{P}_{\text{Si},2}$ must be 2 times the current estimate. Hence, the row of each existing item has to be multiplied by 2: $v[1,1]$ is moved to $v[2,1]$ and $v[3,1]$ is discarded since the table has only 5 rows. Since $\overline{n}_3 = 5$, Δq_3 is divided by 5 and inserted in $v[5,1]$. The new estimate is $\overline{P}_{\mathrm{Si},3} = 0.5 \cdot (v[2,1] + v[5,1]) = 8\,851$. In this iteration, the interesting case occurs, where a value of P_{Si} is estimated which is a divisor of all measurements. This is possible due to the fact that coprime approximate multiples of P_{Si} are measured. The table content after all example measurements is as well shown in Fig. 6.2. Here, the columns of the table are used to keep different measurements of the same approximate multiple, overwriting old measurements.

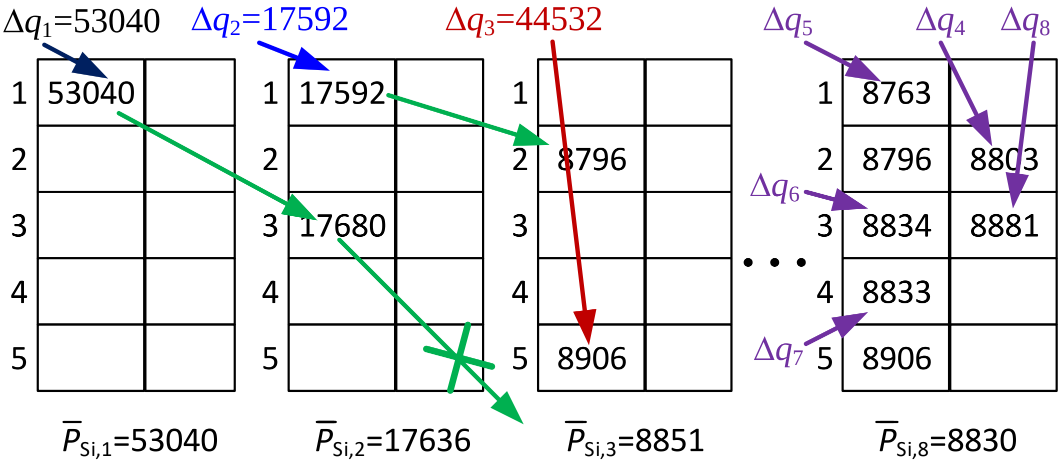

Figure 6.2: Example iterations of Algorithm 7 with table v.

Algorithm 7 formalizes our ACS-PEDC method.

Algorithm 7 keeps the table v that is updated in each iteration j to record S samples of up to L multiples of P_{Si}. Hereby, L in line 3 is the maximum possible value of $\overline{m}_j + \overline{n}_j$ that leads to a valid estimate of the AGCD according to Theorem 2. An additional vector c keeps track of the location of the last recorded sample for each multiple $l \leq L$. All array entries are initialized as zero. Iteration j of Algorithm 7 starts with measuring the current values Δq_j and e_{P}. In the first iteration, Δq_1 is directly written as the first value in v and the estimate of P_{Si} is $\overline{P}_{\mathrm{Si},1} = \Delta q_1$ (line 9). Otherwise, it is checked for all possible combinations of $m, n \leq L$ if (6.4) and (6.5) can be fulfilled for the previous estimate $\overline{P}_{\mathrm{Si},j-1}$ and the current value Δq_j (line 11). In the negative case, the values of iteration $j-1$ are retained (line 21). In the positive case, line 11 can be re-written as

$$\frac{\overline{P}_{\mathrm{Si},j-1}}{\overline{m}_j} - \frac{\Delta q_j}{\overline{n}_j} < \frac{(\overline{m}_j + \overline{n}_j) \cdot X}{\overline{m}_j \cdot \overline{n}_j}.$$

Algorithm 7 ACS-PEDC

```
Input: $X, Y, S$
Output: $\overline{s}_{\mathrm{Si},j}$ and $\overline{P}_{\mathrm{Si},j}$ at each iteration $j$
Init: $j = 0$; $\overline{P}_{\mathrm{Si},j} = 0$; $L = \lfloor \frac{Y}{2X} \rfloor$; $v[l,s] = 0$ and $c[l] = 0$ for $1 \leq l \leq L$ and $1 \leq s \leq S$
while true do
    if $\widehat{\mathrm{PEO}}$ occurs (PEO is detected) then
        $j = j + 1$; measure $\Delta q_j$ and $e_{\mathrm{P}}$
    end if
    if j = 1 then
        $v[1,1] = \Delta q_1$; $c[1] = 1$; $\overline{P}_{\mathrm{Si},1} = \Delta q_1$; $\overline{n}_1 = 1$
    else
        if $\overline{P}_{\mathrm{Si},j-1}$ and $\Delta q_j$ fulfill (6.4), (6.5) for some $\overline{n}_j, \overline{m}_j$ then
            for $l = L \cdots 1$ do
                if $l \cdot \overline{m}_j \leq L$ then
                    $v[l \cdot \overline{m}_j, s] = \lfloor v[l,s]/\overline{m}_j \rceil$ for all $s \leq S$
                    $c[l \cdot \overline{m}_j] = c[l]$
                end if
            end for
            $c[\overline{n}_j] = c[\overline{n}_j] \mod S + 1$
            $v[\overline{n}_j, c[\overline{n}_j]] = \lfloor \Delta q_j / \overline{n}_j \rceil$;
        else
            $\overline{P}_{\mathrm{Si},j} = \overline{P}_{\mathrm{Si},j-1}$
        end if
    end if
    $\overline{P}_{\mathrm{Si},j}$ is the average of all non-zero values in $v$
    $\overline{s}_{\mathrm{Si},j} = \mathrm{sgn}(e_{\mathrm{P}})$
end while
```

That is, the new estimate of P_{Si} will be in the order of $\frac{\overline{P}_{\mathrm{Si},j-1}}{\overline{m}_j} \approx \frac{\Delta q_j}{\overline{n}_j}$. Accordingly, each entry in row l of v_{j-1} (representing multiple l of $\overline{P}_{\mathrm{Si},j-1}$) is divided by $\overline{m}_j$ and moved to row $l \cdot \overline{m}_j$ (representing multiple $l \cdot \overline{m}_j$ of the new estimate) and c is updated accordingly in line 14 and 15. In addition, since $\Delta q_j \approx \frac{\overline{n}_j \cdot \overline{P}_{\mathrm{Si},j-1}}{\overline{m}_j}$, the new sample $\lfloor \Delta q_j / \overline{n}_j \rceil$ is written in row $\overline{n}_j$ of v. After updating v, the estimate $\overline{P}_{\mathrm{Si},j}$ in iteration j is computed as the average of all valid samples (non-zero values) in v and $\overline{s}_{\mathrm{Si},j}$ is determined by the sign of the phase error e_{P}.

We next show a sufficient condition for a tight bound on the difference between P_{Si} and the estimates $\overline{P}_{\mathrm{Si},j}$.

Theorem 3 *(i) It holds at each iteration j of Algorithm 7 with measurement $\Delta q_j = n_j \cdot P_{\mathrm{Si}} + \Delta p_j$ and multiple $\overline{n}_j$ in line (11) that $\overline{P}_{\mathrm{Si},j} = \overline{k}_j \cdot P_{\mathrm{Si}} + \Delta \overline{p}_j$ with $\overline{k}_j = \min_{r=1,\ldots,j}\{\frac{n_r}{\overline{n}_r}\}$ and $|\Delta \overline{p}_j| < X$. (ii) If additionally $\overline{n}_j = n_j$ in line 11 of Algorithm 7 and there is no zero entry in v, it holds for all $r \geq j$ that*

$$|\overline{P}_{\mathrm{Si},r} - P_{\mathrm{Si}}| < X_{\mathrm{eff}} = X \cdot \frac{H_L}{L}. \tag{6.7}$$

Hereby, $H_L = \sum_{n=1}^{L} \frac{1}{n}$ denotes the L-th harmonic number.

The proof of Theorem 3 is given in Appendix B.2. In order to explain the theorem, we recall that all estimates are approximate multiples of P_{Si}. Part (i) of Theorem 3 states that the quality of the estimate of P_{Si} in iteration j is determined by the smallest multiple found up to this iteration. Part (ii) establishes the desired condition of closely estimating P_{Si} (that is, the approximate multiple should be 1). On the one hand, this condition is trivially fulfilled if a measurement $\Delta q_j = 1 \cdot P_{\mathrm{Si}} + \Delta p_j$ with $n_j = 1$ is taken. However, the timely observation of such measurement is only expected if the TM frequently transmits messages as will be further explored in Section 6.5.2. On the other hand, the condition is already fulfilled if coprime multiples of P_{Si} are observed, which is much more likely as shown in Section 6.5.2. It is important to note that X_{eff} in (6.7) denotes the effective uncertainty bound, which is a decreasing function of the number L of multiples. Specifically, the usage of L multiples of P_{Si} has the advantage of improving the estimate of P_{Si}.

Based on Theorem 3, it is further possible to evaluate the clock accuracy of ACS-

PEDC for the case $\overline{n}_j = n_j$ as

$$\max_t\{|c_{\mathrm{Si}}(t)|\} \leq \Delta_{\mathrm{PEO}} = C_{\mathrm{OC}} + \\ \left(\left|\left\lceil\frac{T \cdot X_{\mathrm{eff}}}{\gamma \cdot T_{\mathrm{Si}} \cdot (P_{\mathrm{Si}} - X_{\mathrm{eff}}) \cdot (P_{\mathrm{Si}} - 1)}\right\rceil\right| + 1\right) \cdot \gamma \cdot T_{\mathrm{Si}}. \tag{6.8}$$

The detailed proof of this identity is given in Appendix B.3.

We also emphasize that both the usage of the AGCD for the clock drift estimation and the result in Theorem 2 for the computation of the AGCD are new. Different from the existing literature that performs offline computations [111, 112] or tries to recover the exact value of the AGCD [110, 113] in different application areas, we perform an online computation of the AGCD that suitably updates the drift estimate based on new measurements.

We finally point out that the developed method will as well be applicable for CAN FD and CAN XL due to the usage of the same bit timing mechanism.

6.4 Experimental Evaluation

The main focus of this section is the experimental evaluation of ACS-PEDC for different performance parameters. The same experimental setup that is used while evaluating the RM-based CS method and is introduced in Section 5.4.2 is constructed to evaluate the performance of ACS-PEDC method. Each board realizes a 32-bit LC with $T_N = 250$ ns. Moreover, the FPGA development board [67] (node N4) accommodates our custom CCIP such that the phase error values are accessible different than the existing CAN controllers. Thus, N4 can apply ACS-PEDC in order to realize clock synchronization even though the other nodes (N1-N3) use standard CAN controllers inside the MCU development boards [65].

6.4.1 Implementation Details

The implementation of ACS-PEDC consists of a custom CCIP, a PEO detection block (PEOD), a 32-bit hardware timer block, an interface block and software running on the ARM Cortex-A9 processor of the board as can be seen in Fig. 6.3.

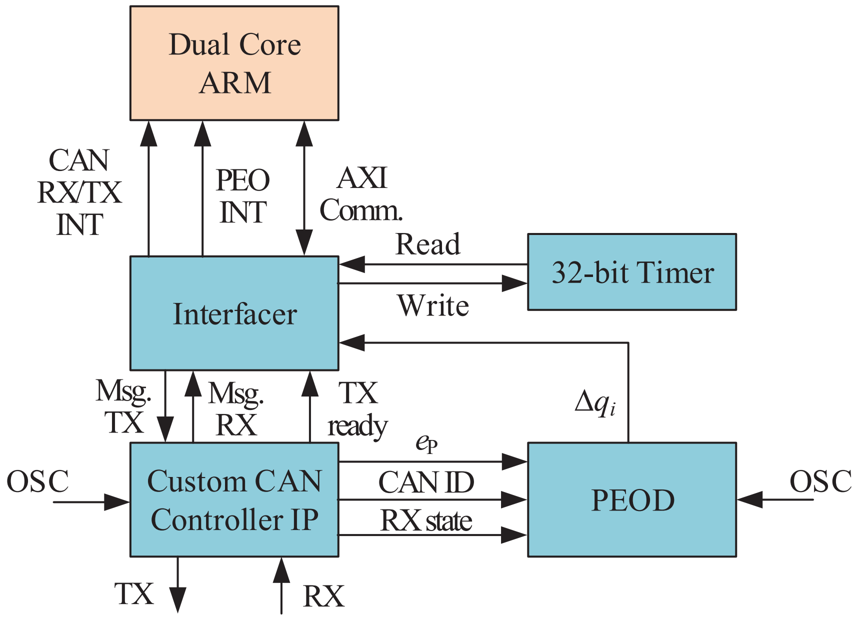

Figure 6.3: The components of ACS-PEDC.

The custom CCIP realizes the standard CAN controller functionality without any modifications. In particular, our custom CCIP performs message TX/RX and provides the related TX/RX flags. Different from a commercial CAN controller, our custom CCIP makes the phase error e_{P}, the current RX message ID and the RX state (current bit received in the CAN frame) available for reading. Here, it is important to note that this does not change the CAN functionality such that a CAN node with the CCIP can operate with standard CAN nodes. This is actually the case in our experimental setup with the standard CAN nodes TM, S1 and S2. Only S3 implements the CCIP. The value of the 32-bit hardware timer can be read for timestamping and incremented/decremented based on the offset correction and the drift correction. PEOD realizes the PEO detection in line 6 of Algorithm 7 during the effective part of TM messages. The effective part is given by bits whose edges are definitely controlled by the TM without interference from other nodes. Since all nodes potentially participate in the arbitration and the acknowledgement (ACK), the effective part of TM messages is given by all bits after the arbitration field and before the ACK field (see white parts in Fig. 2.1 and 2.2). Thus, the first re-synchronization in the effective part is ignored since it may be belong to the first edge that is exactly created from TM. Moreover, the non-zero values of e_{P} during the rest of the effective part of TM messages are considered as valid PEOs by the PEOD block. Hereby, the effective part is determined from the RX state that defines which bit is being received. The Interfacer block realizes the data exchange between the FPGA and the ARM processor. Messages, timer values and clock updates are exchanged via the AXI bus. In addition, the CAN RX/TX interrupts as well as an interrupt for any new Δq_j measurement are provided. Using these interrupts, the ARM processor polls the relevant data via the AXI bus. All parts of ACS-PEDC except for the PEO detection and Δq_j measurement are realized on the ARM processor. The resulting clock update values are then applied to the hardware timer.

Remark 8 *We note that commercial CAN controllers do not provide the phase error information. However, since the phase error is computed in each CAN controller, only a minor modification of existing CAN controllers is required in order to make the phase error available. Most importantly, this modification does not change the operation of the CAN protocol.*

6.4.2 Comprehensive Experimental Evaluation

Using the described implementation of ACS-PEDC, we next perform a comprehensive experimental evaluation. In this context, we introduce the bus utilization U as the number of bits per time unit sent by all nodes divided by the bit rate. Similarly, U_{TM} denotes the bus utilization of the TM.

Then, we investigate the dependency of the obtained clock accuracy on relevant parameters such as the RM period T, the bit rate B, U_{TM}, U, γ, the clock drift of the slave node d_{Si} and environmental influences such as the temperature. Each measurement was conducted for a time duration of 1 800 s, which corresponds to a number of $K = 90\,000$ LC measurements. The values of T and bit rates used in practice are chosen for the experiments. Our measurement results are summarized in Table 6.1.

We first evaluate the dependency on the RM period. Representative measurements are shown in Fig. 6.4. It is readily observed that the absolute clock differences are well below $2\,\mu$s with sporadic outliers. In order to quantify this observation, we write $K_{\geq\delta}$ for the number of samples with a clock difference above a given bound δ. Then,

Table 6.1: Comparison of Δc_{S3} for different experiments.

	mean [μs]	max [μs]	std [μs]	$p_{\mathrm{O},2\,\mu\mathrm{s}}$ [%]
Dep. on T [s]: $B = 250\,$kbps, $U_{\mathrm{TM}} = 30\%$, $\gamma = 4$				
$T = 0.262$	0.46	2.5	0.31	0.002
$T = 0.524$	0.43	2.3	0.30	0.002
$T = 1.048$	0.33	2.5	0.33	0.001
$T = 8.384$	0.46	2.0	0.32	0.0
$T = 16.768$	0.36	2.5	0.27	0.002
Dep. on B [kbps]: $T = 1.048\,$s, $U_{\mathrm{TM}} = 30\%$, $\gamma = 4$				
$B = 500$	0.56	2.8	0.33	0.003
$B = 1000$	0.61	2.5	0.33	0.003
Dep. on U_{TM} [%]: $T = 1.048\,$s, $B = 250\,$kbps, $\gamma = 4$				
$U_{\mathrm{TM}} = 10\%$	0.58	3.0	0.34	0.002
$U_{\mathrm{TM}} = 50\%$	0.60	2.75	0.33	0.003
Dep. on U [%]: $T = 1.048\,$s, $B = 250\,$kbps, $\gamma = 4$, $U_{TM} = 10\,\%$				
$U = 20\%$	0.60	2.8	0.33	0.003
$U = 50\%$	0.50	2.5	0.33	0.004
$U = 70\%$	0.55	2.8	0.32	0.003
$U = 85\%$	0.49	2.8	0.32	0.005
Dep. on γ: $T = 1.048\,$s, $B = 250\,$kbps, $U_{\mathrm{TM}} = 30\%$				
$\gamma = 2$	0.27	2.3	0.21	0.003
$\gamma = 1$	0.12	2.0	0.14	0.0
Dep. on $d_{\mathrm{S3}}(t)$: $T = 1.048\,$s, $B = 250\,$kbps, $U_{\mathrm{TM}} = 30\%$, $\gamma = 4$				
$\overline{d}_{\mathrm{S3}} \approx -78$ppm	0.50	2.75	0.32	0.005
$\overline{d}_{\mathrm{S3}} \approx 8$ppm	0.52	1.75	0.33	0.0
Dep. on Temp.: $T = 1.048\,$s, $B = 250\,$kbps, $U_{\mathrm{TM}} = 30\%$, $\gamma = 4$				
	0.40	2.5	0.31	0.004

we define the percentage of outliers as

$$p_{\mathrm{O},\delta} = \frac{K_{\geq\delta}}{K} \cdot 100. \tag{6.9}$$

Table 6.1 shows that the percentage of outliers is negligible. Most importantly, ACS-PEDC achieves the same accuracy below $2\,\mu$s even for very infrequent RMs due to the very accurate drift estimate in the order of ± 0.1 ppm as can be seen in the right-hand plots of Fig. 6.4. That is, the drift estimates when using ACS-PEDC are one order of magnitude better compared to drift estimates that are computed from RMs when comparing the measurements in Fig. 5.12.

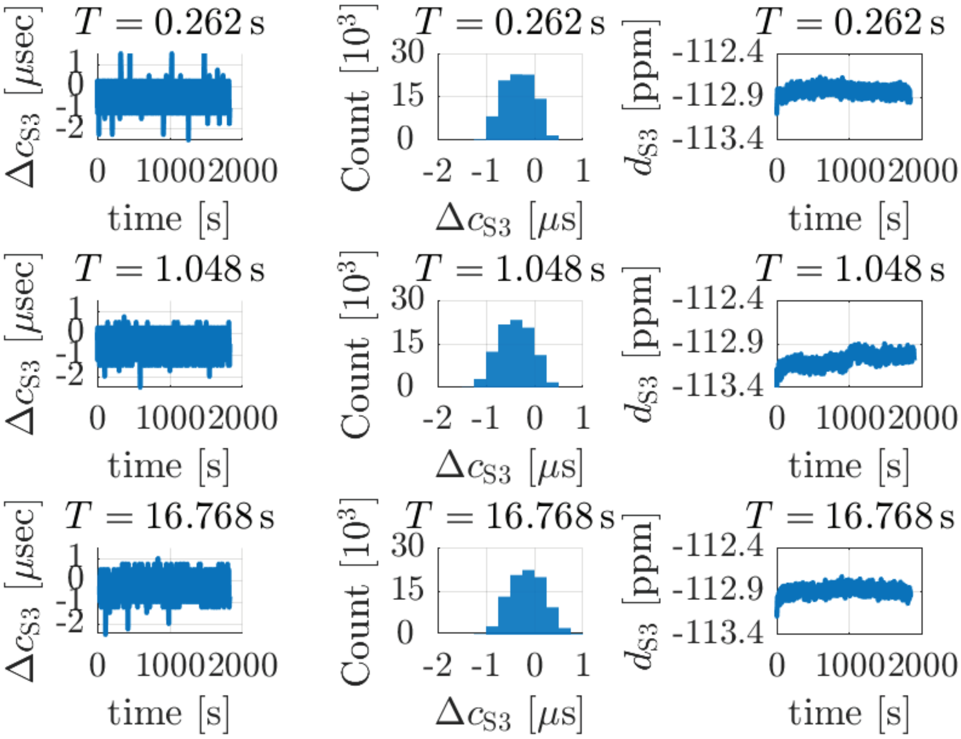

Figure 6.4: Measurements for different RM periods.

Remark 9 *Existing fault-tolerant CS methods for CAN experience a deterioration of the clock accuracy during fault-recovery [43] since they are based on RMs only. Since ACS-PEDC does not require frequent RMs and hence tolerates the intermittent loss of several RMs, it is expected to improve the clock accuracy during fault-recovery. This advantage of ACS-PEDC will be further investigated in future work.*

Remark 10 *We also point out that the clock difference when using ACS-PEDC is mostly affected by the offset correction in* (5.1) *due to the software inaccuracies. That is, we expect an improved clock accuracy when using a hardware implementation of the timestamping service, which is the topic of our ongoing research.*

For the next comparisons, we note that the measurement for $B = 250\,\text{kbps}$, $T = 1.048\,\text{s}$, $U_{\text{TM}} = 30\%$ and $\gamma = 4$ already appears in the upper part of the Table (highlighted in gray) and is hence not repeated.

Table 6.1 shows that the clock accuracy does not depend on the bit rate B when using ACS-PEDC. As pointed out in Section 6.4.1, PEO can only be detected during the effective part of any message transmitted by the TM. Hence, it is expected that more PEOs are missed if the bandwidth U_{TM} used by the TM for message transmissions is low. In order to study this effect, we conduct an experiment with different values of $U_{\text{TM}} = 10\%, 30\%, 50\%$ as shown in Table 6.1. Here, the interesting result is that the same clock accuracy is maintained even for a small value of $U_{\text{TM}} = 10\%$. In addition, it holds that there is no interference by transmissions of other nodes, such that the clock accuracy as well does not depend on an increasing overall bus utilization U with more transmissions from other nodes as can be seen in Table 6.1.

The parameter γ determines the correction period for clock updates, which is realized by counting $\gamma \cdot \overline{P}_{\text{S3}}$ HR ticks by S3. Noting that the previous experiments were carried out for $\gamma = 4$ and hence, $\gamma \cdot \overline{P}_{\text{S3}} \cdot T_{Si} \approx 8.8\,\text{ms}$, we conducted experiments for $\gamma = 2$ and $\gamma = 1$ as shown in Table 6.1. It can be seen that decreasing γ leads to smaller clock differences as is expected from (6.8). Accordingly, it is possible to further improve the clock accuracy by selecting small values of γ at the expense of an increased load for the ARM processor.

The previous experiments were carried out using N1 as the TM and the slave node S3 = N4. Accordingly, the obtained drift estimates are all similar in the order of $-113.1\,\text{ppm}$ with small variations depending on the environmental conditions. Table 6.1 shows measurements for experiments where the clock oscillator of the TM was changed to obtain drift values of $d_{\text{S3}}(t) \approx -78\,\text{ppm}$ and $d_{\text{S3}}(t) \approx 8\,\text{ppm}$. It can be seen that there is no significant effect of the clock drift on the clock accuracy. This is expected since the clock accuracy is dominated by the offset correction, whereas ACS-PEDC is guaranteed to work as long as all the parameters remain within the defined bounds.

All the previous experiments were conducted with similar environmental conditions such that no significant variations in the clock drift are observed (see also Section

2.2.1). We next investigate the performance of ACS-PEDC under a time-varying clock drift. To this end, we exploit the dependency of the clock drift on the temperature and repeatedly increase and decrease the ambient temperature of the TM with the following profile: heat up ($60-120\,\mathrm{s}$), cool down ($120-240\,\mathrm{s}$), heat up ($240-360\,\mathrm{s}$), cool down ($360-660\,\mathrm{s}$), heat up ($660-720\,\mathrm{s}$), cool down ($720-1200\,\mathrm{s}$), heat up ($1200-1320\,\mathrm{s}$), cool down ($1320\,\mathrm{s}$ to end).

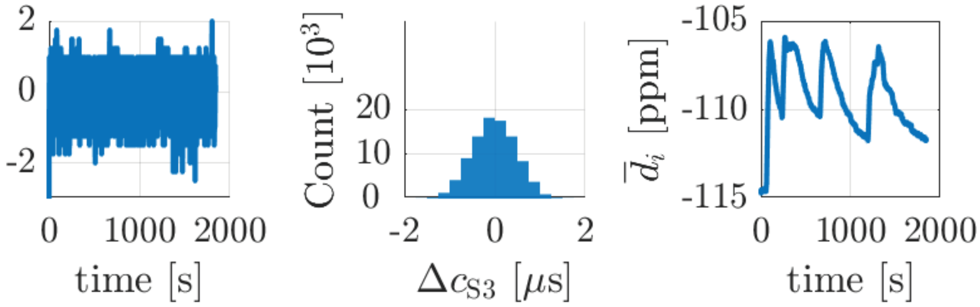

Figure 6.5: Measurements for a time-varying clock drift.

The measurement in Fig. 6.5 shows that ACS-PEDC precisely and quickly adjusts to a time-varying drift. This is due to the fact that the algorithm only keeps a limited number of measurements that are updated with each new value Δq_j in the array v_i. Accordingly, there is also no effect on the clock accuracy. We further note that the experiment represents a very fast change of the TM temperature which is not expected in practice.

We finally evaluate the implementation complexity of ACS-PEDC in comparison to the RM-based CS method with drift correction in Section 5.4.1. Using $\gamma = 4$, we determined a processor utilization of $0.27\,\%$ and $0.42\,\%$ for AUTOSAR-DC and ACS-PEDC, respectively. It is readily observed that the processor utilization is negligible, which confirms the practicality of the proposed method.

6.4.3 Start-up Phase

The tight bound in (6.8) is ensured as soon as coprime multiples of P_{Si} are observed according to Theorem 3. We define this time as the PEO period detection time $t_{\mathrm{PEO,Si}}$.

We next compare the PEO period detection times for the naïve approach in Section

6.1, which only considers the smallest Δq_j measurements and ACS-PEDC. The results for the maximum and mean value of $t_{\mathrm{PEO,S3}}$ for different levels of U_{TM} are summarized in Table 6.2 (results for 20 trials per data point are shown).

Table 6.2: Maximum and mean PEO period detection times.

	Maximum $t_{\mathrm{PEO,S3}}$			Mean $t_{\mathrm{PEO,S3}}$		
U_{TM}	$10\,\%$	$30\,\%$	$50\,\%$	$10\,\%$	$30\,\%$	$50\,\%$
Naïve [s]	413.2	8.65	0.70	139.8	2.21	0.16
Algorithm 7 [s]	207.9	3.42	0.23	43.0	0.68	0.09

The first important observation is that the naïve method has larger PEO period detection times than ACS-PEDC. This is expected since ACS-PEDC already determines the correct PEO period if coprime multiples of $P_{\mathrm{DC,S3}}$ are measured. However, it is still the case that the PEO period detection times can be large for low values of U_{TM} since it is less likely to quickly observe coprime multiples of $P_{\mathrm{DC,S3}}$ in the PEO measurements. This is a possible disadvantage of ACS-PEDC in practical applications since it is not guaranteed to achieve a tight drift estimate short after system start-up. This issue is addressed in the next section.

6.5 Start-up Method for ACS-PEDC

The experiments in Section 6.4 indicate that ACS-PEDC enables highly accurate CS for different RM periods, bit rates, bus utilization, clock drifts and temperatures without modifying the CAN protocol. As pointed out in Section 6.4.3, a possible shortcoming of ACS-PEDC is the fact that a close estimate of P_{Si} is only found after measuring coprime multiples of P_{Si} after system start-up. We next develop a method for achieving short start-up times by combining information from Algorithm 7 and the RM timestamps.

6.5.1 Start-up Method

We first determine bounds for a drift estimate based on RM timestamps. To this end, we consider the case, where no offset correction and drift correction is applied to the clock c_{Si} after the first RM. That is, ideally, $N_{\mathrm{Si}}(T) = \lfloor T/T_{\mathrm{Si}} \rfloor$. Assuming an approximately constant drift d_{Si} during the start-up period, d_{Si} can be evaluated after the k-th RM ($k \geq 2$) as

$$d_{\mathrm{Si},k}^{\mathrm{RM}} = \frac{(\hat{N}_{\mathrm{Si},k} - \lfloor \hat{t}_{\mathrm{TM},k}/T_{\mathrm{Si}} \rfloor) \cdot T_{\mathrm{Si}}}{(k-1) \cdot T}. \tag{6.10}$$

It has to be respected that the timestamps $\hat{N}_{\mathrm{Si},k}$ (Si) and $\hat{t}_{\mathrm{TM},k}$ (TM) are not ideal and there is a propagation delay on the CAN bus. Using (6.10) and the maximum clock correction error of Si

$$E_{\mathrm{Si}}^{\mathrm{CC}} = \left(\left\lceil \frac{E_{\mathrm{TRI}} + J_{\mathrm{Si}}}{T_{\mathrm{Si}}/(1 + d_{\mathrm{Si}})} \right\rceil + \left\lceil \frac{J_{\mathrm{TM}}}{T_{\mathrm{Si}}} \right\rceil + \left\lceil \frac{E_{\mathrm{TRI}}}{T_{\mathrm{Si}}/(1 + d_{\mathrm{Si}})} \right\rceil + 1 \right) \cdot T_{\mathrm{Si}}, \tag{6.11}$$

the following upper and lower bounds for the actual drift can be determined[2].

$$d_{\mathrm{Si},k}^{\mathrm{lo}} = d_{\mathrm{Si},k}^{\mathrm{RM}} - \frac{E_{\mathrm{Si}}^{\mathrm{CC}}}{(k-1) \cdot T} \leq d_{\mathrm{Si}} \leq d_{\mathrm{Si},k}^{\mathrm{up}} = d_{\mathrm{Si},k}^{\mathrm{RM}} + \frac{E_{\mathrm{Si}}^{\mathrm{CC}}}{(k-1) \cdot T}. \tag{6.12}$$

Hereby, it is important to note that the bounds in (6.12) become tighter (the drift estimate becomes more accurate) with increasing values of k.

In order to compare the bounds in (6.12) to the drift estimate from Algorithm 7, we next recall the fact that the actual value P_{Si} always must be an approximate divisor of $\overline{P}_{\mathrm{Si},j}$. That is, $\overline{P}_{\mathrm{Si},j} = \overline{k}_j \cdot P_{\mathrm{Si}} + \Delta \overline{p}_j$ with $|\Delta \overline{p}_j| \leq X$. Accordingly, we propose to evaluate drift estimates from divisors of $\overline{P}_{\mathrm{Si},j}$ in the form

$$\overline{d}_{\mathrm{Si},j}(m) = \frac{s_{\mathrm{Si}}}{\overline{P}_{\mathrm{Si},j}/m - s_{\mathrm{Si}}} = \frac{s_{\mathrm{Si}} \cdot m}{\overline{P}_{\mathrm{Si},j} - s_{\mathrm{Si}} \cdot m} \tag{6.13}$$

in order to determine $\overline{k}_j$. Taking into account the possible uncertainty X in the estimate $\overline{P}_{\mathrm{Si},j}$, the actual drift for divisor m is bounded by

$$\begin{aligned} d_{\mathrm{Si},j}^{\mathrm{lo}}(m) = \frac{s_{\mathrm{Si}} \cdot m}{\overline{P}_{\mathrm{Si},j} + s_{\mathrm{Si}} \cdot X - s_{\mathrm{Si}} \cdot m} &\leq \overline{d}_{\mathrm{Si},j}(m) \\ \leq \frac{s_{\mathrm{Si}} \cdot m}{\overline{P}_{\mathrm{Si},j} - s_{\mathrm{Si}} \cdot X - s_{\mathrm{Si}} \cdot m} &= d_{\mathrm{Si},j}^{\mathrm{up}}(m). \end{aligned} \tag{6.14}$$

---

[2] Appendix B.4 provides the derivation of the results in this section.

In view of (6.12) and (6.14), the drift estimate in (6.13) is considered as safe if

$$d_{\mathrm{Si},j}^{\mathrm{up}}(m-1) < d_{\mathrm{Si},k}^{\mathrm{lo}} \text{ and } d_{\mathrm{Si},j}^{\mathrm{lo}}(m+1) > d_{\mathrm{Si},k}^{\mathrm{up}}. \tag{6.15}$$

In this case, the drift estimate from the PEO measurement is guaranteed to respect the bounds computed from the RM timestamps for a unique value of m. Hence, it must be the case that $m = \overline{k}_j$.

We note that the conditions in (6.15) solely depend on measurements and parameters that are available on Si. That is, Si can evaluate these conditions to decide about the correctness of the drift estimate as summarized in Algorithm 8.

Algorithm 8 Start-up procedure for ACS-PEDC

```
 1: Init: active = START-UP
 2: while true do
 3:     Determine $\overline{P}_{\mathrm{Si},j}$ using Algorithm 7
 4:     if $k < 2$ then
 5:         Apply drift correction using $\overline{P}_{\mathrm{Si},j}$
 6:     else
 7:         Evaluate the conditions in (6.15)
 8:         if there is a unique value of $m$ then
 9:             active = ACS-PEDC
10:         end if
11:         if active = ACS-PEDC then
12:             Apply drift correction using $\overline{P}_{\mathrm{Si},j}/m$
13:         else
14:             Apply drift correction using $P_{\mathrm{Si}}^{\mathrm{RM}} = s_{\mathrm{Si}} + s_{\mathrm{Si}}/d_{\mathrm{Si},k}^{\mathrm{RM}}$
15:         end if
16:     end if
17: end while
```

In the start-up phase, Algorithm 8 uses the less accurate drift estimate from the RM timestamps (line 14) and checks the conditions in (6.15). If a unique value of m is found, the start-up phase is completed and the highly accurate drift estimate from the PEO measurements is used (line 12).

We next show that the proposed method ensures a fast start-up independent of the drift value.

Theorem 4 *Consider Algorithm 8 and write* t_{ST} *for the start-up time, when* `active` = ACS-PEDC. *Then,*

$$\max_t\{|c_{\mathrm{Si}}(t)|\} \leq \begin{cases} \Delta_{\mathrm{RM}} = C_{\mathrm{OC}} + \frac{E_{\mathrm{Si}}^{\mathrm{CC}}}{(k-1)} & \textit{if}\, T \leq t \leq t_{\mathrm{ST}} \\ \Delta_{\mathrm{PEO}}\ \textit{in}\ (6.8) & \textit{if}\, t > t_{\mathrm{ST}}. \end{cases} \tag{6.16}$$

Remark 11 *The drift estimate in* (6.10) *is based on a clock value of* c_{Si} *without offset correction and drift correction. When applying offset correction using the AUTOSAR method and drift correction according to line 14 in Algorithm 8, the clock value without offset correction and drift correction can be obtained by memorizing the accumulated offset correction and drift correction during the start-up period.*

Remark 12 *The bound in* (6.16) *converges to* C_{OC} *for large values of* k*. It has to be pointed out that this bound is only considered at system start-up since it relies on the assumption that the actual drift* d_{Si} *is constant. After the start-up phase, Algorithm 7 enables highly accurate drift estimates even for time-varying drifts as is shown in Fig. 6.5.*

6.5.2 Start-up Evaluation

We next evaluate the start-up phase for different drifts and TM bus loads U_{TM}. Fig. 6.6 shows representative measurements of the clock difference of S3 for $U_{\mathrm{TM}} = 10\%$, $B = 250\,$kbps, $T = 1.048\,$s, $\gamma = 4$, $d_{\mathrm{Si}} = 8\,$ppm and $d_{\mathrm{Si}} = -113\,$ppm. Hereby, we recall that the first offset correction of S3 is performed with the reception of the first RM, which is assumed to occur at $t = 0$ in the figure for ease of understanding. In general, a slave node will receive the first RM between $t = 0$ and $t = T$.

It is readily observed that c_{S3} drifts freely until the second RM at around $t = T$. After that, Δc_{S3} stays below the bound $\pm\Delta_{\mathrm{RM}}$ in (6.16) until t_{ST} (represented by the dashed line). Finally, Δc_{S3} is bounded by $\pm\Delta_{\mathrm{PEO}}$ in (6.8). Hereby, it has to be further noted that the bounds consider the worst case, which was not observed in any of our experiments. In order to quantify this observation, we define the time $t_{2\,\mu\mathrm{s}}$ as the time

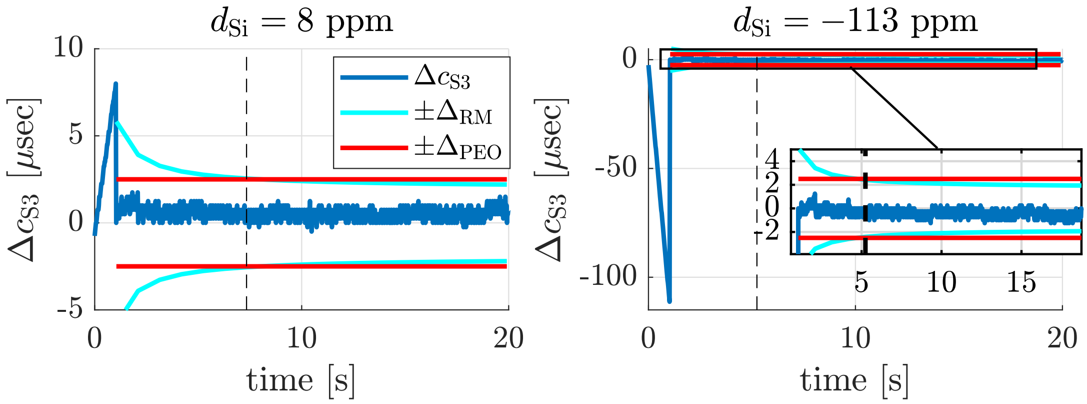

Figure 6.6: Start-up measurements for different drift values.

Table 6.3: Evaluation of $t_{2\,\mu\mathrm{s}}$ depending on U_{TM} and d_{S3}.

U_{TM}	6.5%	10%	30%	50%
$d_{\mathrm{S3}} = 8\,\mathrm{ppm}$	0.73 s	0.88 s	0.64 s	0.16 s
$d_{\mathrm{S3}} = -78\,\mathrm{ppm}$	0.73 s	0.84	0.0	0.0
$d_{\mathrm{S3}} = -113.1\,\mathrm{ppm}$	0.72 s	0.63	0.01	0.0

after which the clock difference of S3 is below $2\,\mu$s. Table 6.3 shows the average value of $t_{2\,\mu\mathrm{s}}$ over 10 experiment runs for different values of U_{TM} and d_{S3}.

Our experiments indicate that a clock accuracy below $2\,\mu$s is generally achieved within less than 1 s. In particular, we observed that this clock accuracy could always be achieved after the slave receives the second RM. We emphasize that the proposed CS method is applicable even using TMs with a very small bus utilization.

6.6 Discussion and Conclusion

This chapter proposes a new CS algorithm for CAN that combines infrequent clock updates using timestamps in RMs from a TM and a highly precise drift correction between such clock updates. The original idea of our algorithm is the measurement of the time intervals between re-synchronizations of the CAN bit time. We develop a novel drift estimation algorithm based on the real-time computation of the AGCD of these time intervals and formally prove its correctness. We further extend our algorithm by a method for the quick start-up of the CS. In order to validate the practicability of the proposed method, we conduct extensive experiments.

Generally, the experiments show that any CAN node that implements our method is able to estimate the clock drift with a precision below 0.2 ppm and achieves a clock accuracy below $2\,\mu$s after receiving at most two RMs from the TM. Hereby, these values are obtained independent of the bit rate, the RM period and the drift between clock oscillators of the nodes. Even time-varying oscillator drifts that can be caused by variations in the ambient temperature do not affect the clock accuracy.

As a further advantage of the proposed method, no modification of the CAN protocol is required. That is, different from the TTCAN protocol, any node implementing our method can be connected to any standard CAN network. Different from other methods that employ drift correction based on timestamps only, our method achieves a highly accurate drift estimate using the bit time synchronization of CAN. Thus, CANDS framework can follow our novel ACS-PEDC as a clock synchronization method. Moreover, ACS-PEDC enables estimating the clock drift independently from the periodic timestamps within RMs which has advantages when timestamping quality is low. When our ACS-PEDC is compared to the CS methods which benefits from the timestamps both for offset and drift correction, the performance of ACS-PEDC is degraded less by timestamping quality since it depends on TS quality only for offset correction, not for drift correction.

CHAPTER 7

WEAK TDMA ON CAN

The proposed CS algorithms that can also apply drift correction for CAN in Chapter 5 provide clock accuracy below 5 μs independently from the clock drift of the nodes. In Chapter 4, a new timestamping method is introduced and it is shown in Chapter 5 that it improves the clock accuracy performance of the CS methods by enabling higher quality timestamps. Additional to the timestamp-dependent drift estimate methods, a novel drift computation method that benefits from the internal bit timing mechanism of the CAN protocol is proposed in Chapter 6. Moreover, a new CS method which combines the AUTOSAR compatible offset correction and drift correction depend on the novel drift computation is introduced. It is experimentally shown that it provides the clock accuracy below 2 μs independently from the clock drift and even under changing environmental conditions.

Considering the maximum CAN bus speed that is 1 Mbps, the minimum bit rate equals to 1 μs. Thus, a global clock with an accuracy in the order of 5 μs makes possible to apply classical TDMA schedule with the guard times that handles the clock inaccuracy between the nodes. However, it is noticed in this thesis that the specific properties of CAN bus remove the necessity of the guard times that decrease the bandwidth efficiency. Thus, a new WTDMA model for CAN is introduced in this chapter. Specifically, the WTDMA model defines the traffic shaping for CANDS which provides that each message is transmitted in its own scheduled window.

The WTDMA on CAN is based on two main observations. First, it holds that CAN is a non-preemptive bus and possible message collisions are managed by the CAN arbitration mechanism without a performance penalty. That is, when realizing slot-based access on CAN it is not required to introduce guard times since possible overlaps of time slots will be resolved by the usage of Carrier Sense Multiple Access/Collision Resolution (CSMA/CR) and the non-preemptive message transmission on CAN. Second, it is possible to use CS to keep the overlaps of time slots bounded and hence maintain the transmission order defined by a TDMA schedule that is determined offline.

Based on these observations, we propose a novel WTDMA model for CAN, where time slots can temporarily overlap, different from the classical TDMA. Thus, WTDMA can operate even with a moderate clock accuracy among the nodes that can easily be provided with any of the proposed CS methods for CAN in Chapter 5 and Chapter 6. In addition to the formal description of the WTDMA model, we also validate its usability with comprehensive experiments, where challenging message sets are used. Specifically, we show experimentally that WTDMA operates well even with average bus loads above 95% throughout the experiments with the message sets constructed in line with practical CAN networks. Moreover, we confirm that WTDMA is applicable for all bit rates of CAN and show that our WTDMA model even provides feasible schedules for message sets that are not schedulable on TTCAN. As such, the repetition period of a trigger does not have to be power of 2 and the number of triggers is not limited in WTDMA since it is implemented in software.

In summary, WTDMA which is not limited to any specific CS method, does not require a modification of the CAN protocol and provides very low WCRTs in the order of the message transmission time independently from the assigned message priorities and message deadlines. Hence, WTDMA outperforms all other approaches in the literature in terms of the achieved bus utilization and its implementation complexity.

7.1 Assumptions

We next state three realistic assumptions that are employed in this chapter. First, we note that the CAN protocol itself does not support clock synchronization (CS). Nonetheless, this thesis suggests several CS algorithms that enable an accurate common clock among the nodes without requiring any hardware modification in Chapter 5 and Chapter 6. In this chapter, we assume the availability of a CS algorithm with an accuracy that is bounded by a value θ. That is, any of the existing SW-CS methods can be used as long as the accuracy is guaranteed to be below θ. Secondly, we assume that the software delay of each node $N \in \mathcal{N}$ between the time instant when the generation of a CAN message is triggered and the time instant when it is ready for arbitration is limited by an upper bound Δ_N. Even though the actual delays vary depending on the software implementation, an upper bound can be either estimated experimentally or computed theoretically by examining the hardware on which the software runs. Finally, we point out that messages on CAN are commonly re-transmitted in case of an error. We assume that re-transmission of messages is not allowed similar to TTCAN [100]. It has to be noted that this is not a practical restriction for existing CAN controllers since they can be configured to disable the auto re-transmission feature [114].

7.2 Protocol Definition

As indicated in Section 7.1, we assume that the nodes on a CAN bus have synchronized clocks with a maximum clock difference of θ between any node and the global clock of a TM node.

Then, the proposed medium-access method introduces an offset o_i and a transmission window size w_i for each message $M_i \in \mathcal{M}$. Hereby, the a-th instance of message $M_i \in \mathcal{M}$ is denoted as $M_{i,a}$. Writing $y_{i,a} = o_i + a \cdot p_i$ for $a = 0, 1, \ldots$, we define the time intervals

$$\mathcal{I}_{i,a} = y_{i,a} + [0, w_i), \tag{7.1}$$

during which the message instances $M_{i,a}$ can be transmitted. Accordingly, a message schedule consists of the set of offsets $\mathcal{O} = \{o_1, \ldots, o_m\}$ and the set of window sizes $\mathcal{W} = \{w_1, \ldots, w_m\}$. We further use the hyper-period (HP)

$$H = \operatorname{lcm}(p_1, \ldots, p_m), \tag{7.2}$$

which characterizes the time after which the schedule repeats. The number of repetitions of M_i per HP is $r_i = H/p_i$.

Depending on the choice of $\mathcal{O}$ and $\mathcal{W}$, it is possible that the windows of different messages overlap. Hence, we introduce the notion of a feasible schedule in Definition 1.

Definition 1 *Consider a message set $\mathcal{M}$ with the previously defined parameters and a corresponding schedule given by the set of offsets $\mathcal{O}$ and the set of window sizes $\mathcal{W}$. The schedule is denoted as feasible if*

1. $\forall i \in \{1, \dots, m\}$, *it holds that* $L_i^{\max} \leq w_i$,
2. $\forall i, j \in \{1, \dots, m\}$ *with* $i \neq j$, $a \in \{0, \dots, r_i - 1\}$ *and* $b \in \{0, \dots, r_j - 1\}$, *it holds that* $\mathcal{I}_{i,a} \cap \mathcal{I}_{j,b} = \emptyset$

That is, a feasible schedule ensures that 1) each message fits in its assigned window and 2) the windows of any two different messages do not overlap. That is, each message obtains exclusive access to the CAN bus during its assigned window, ensuring deterministic message transmission.

It has to be noted that this desired property is valid for the ideal case with perfect SW-CS and zero delay between trigger of message generation and transmission start time on the CAN bus. The real case with SW-CS inaccuracies and software delays is addressed in the next section.

7.3 Sufficient Conditions for Correct Operation of WTDMA

The feasible WTDMA schedule in Definition 1 assumes that each message instance is generated and enters arbitration on the CAN bus precisely at the beginning of its allocated window. In practice, it is expected to observe software delays on the CAN nodes as well as deviations of the synchronized clocks.

This section provides an analytical derivation of sufficient conditions for the correct WTDMA operation in the sense that all messages are transmitted in the order defined by the WTDMA schedule. In addition, we determine the WCRT of each message in a given set $\mathcal{M}$.

We first introduce some additional notation for our derivation, which is given for the duration of one HP. Specifically, the following time instants are needed:

- $y_{i,a}$: starting time of the window for $M_{i,a}$,

- $g_{i,a}$: time instant when the generation of $M_{i,a}$ is triggered by its node $\mu(M_i)$,
- $e_{i,a}$: time instant when $M_{i,a}$ is ready to enter arbitration on the CAN bus,
- $u_{i,a}$: time instant when the CAN bus is guaranteed to be free for $M_{i,a}$,
- $s_{i,a}$: time instant when $M_{i,a}$ starts transmission,
- $f_{i,a}$: time instant when $M_{i,a}$ finishes transmission including Inter Frame Spacing (IFS),
- $L_{i,a}$: length of the a-th instance of message M_i.

In addition, we recall the bound $\Delta_{\mu(M_i)}$ for the (MCU-dependent) maximum software delay between the time instant when the generation of message instance $M_{i,a}$ is triggered and the time instant when $M_{i,a}$ is ready to enter arbitration on the CAN bus. Specifically, it consists of the software tasks such as detecting the trigger, preparation of the CAN message content and its placement into the CAN controller transmission buffer. Furthermore, θ is the bound for the clock difference of any node relative to the global clock on the CAN bus. Accordingly, we establish the following facts about the above parameters:

F1 $y_{i,a} - \theta \leq g_{i,a} \leq y_{i,a} + \theta$ and $y_{i,a} - \theta \leq e_{i,a} \leq y_{i,a} + \theta + \Delta_{\mu(M_i)}$,

F2 $s_{i,a} = \max\{u_{i,a}, e_{i,a}\}$,

F3 $f_{i,a} = s_{i,a} + L_{i,a}$.

Fig. 7.1 provides an illustration of the WTDMA operation and the notation introduced above. To this end, the figure shows the transmission of different instances of the consecutive messages M_i, M_j and M_k in two different HPs (denoted as HP_n and HP_{n+1}). The window start times are shown by dashed lines and the range of times where each message instance can enter arbitration is displayed by red and green arrows. Respecting F1, we note that this range is for example given by the interval $[y_{i,a} - \theta, y_{i,a} + \theta + \Delta_{\mu(M_i)}]$ for the message instance $M_{i,a}$. To point out the possible cases, $M_{i,a}$ starts transmission at the latest possible time $s_{i,a} = y_{i,a} + \theta + \Delta_{\mu(M_i)}$ in HP_n of the example. Then, $M_{j,b}$ starts transmission when the bus becomes idle right after the transmission of $M_{i,a}$ at time $s_{j,b} = s_{i,a} + L_{i,a} = f_{i,a} = u_{j,b}$. Here, it is important to note that, even if $e_{j,b} < f_{i,a}$, that is, $M_{j,b}$ is ready before the transmission of $M_{i,a}$ is completed, $M_{j,b}$ is transmitted after $M_{i,a}$ as long as $e_{j,b} > s_{i,a}$. Intuitively, this amounts to the fact that the time instant $s_{i,a}$ when $M_{i,a}$ starts transmission is guaranteed to be before the time instant $s_{j,b}$ when $M_{j,b}$ starts transmission, which will be formally stated and proved in Theorem 5. Similarly, $M_{k,c}$ starts transmission after the transmission of $M_{j,b}$ is completed at $f_{j,b}$. A different scenario is shown for HP_{n+1}. Here, the message instance $M_{i,a+r_i}$ is transmitted at the earliest possible time $y_{i,a+r_i} - \theta$. As a consequence, the CAN bus is idle at $u_{j,b+r_j} = f_{i,a+r_i}$ before $M_{j,b+r_j}$ is ready and $M_{j,b+r_j}$ is transmitted right at $s_{j,b+r_j} = e_{j,b+r_j}$. The same is true for $M_{k,c+r_k}$. At this point, we emphasize that a message can be ready during the time window of the previous message without affecting the WTDMA operation. That is, different from TDMA, the WTDMA operation does not require guard times between time windows for different messages and hence allows a more efficient bandwidth usage.

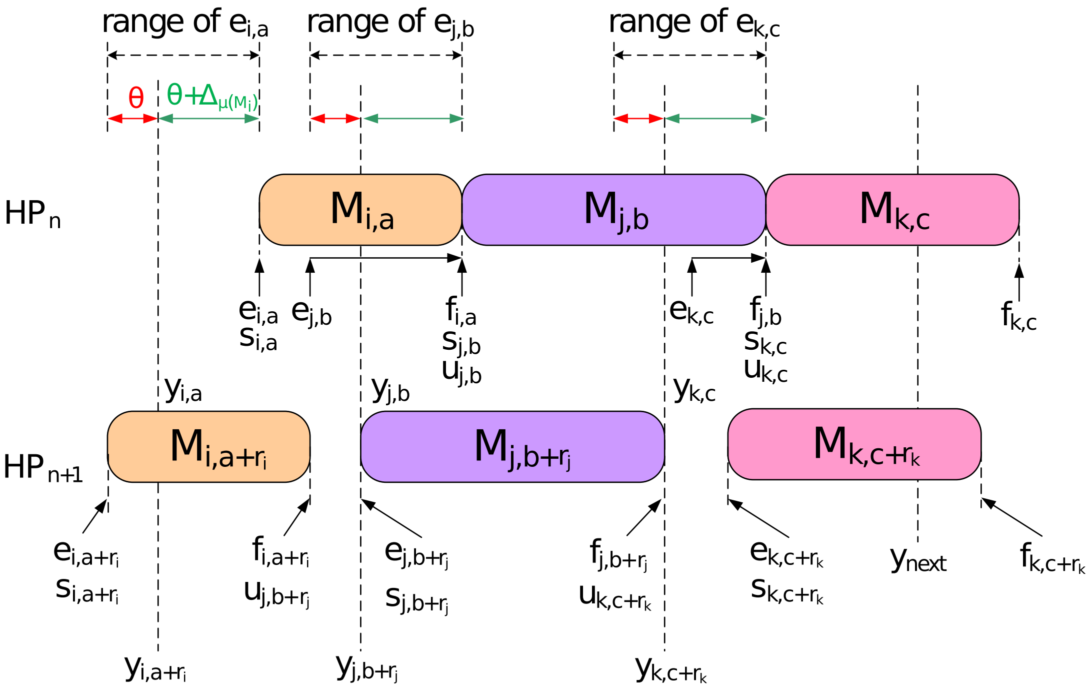

Figure 7.1: WTDMA notation and example illustration.

Using the above notation, we next quantify the correct operation of WTDMA. Consider that K messages are transmitted on the CAN bus according to a given WTDMA schedule. Then, we denote each message that is not transmitted in the specified order as a schedule violation and write

$$V = \frac{\text{Number of schedule violations}}{K} \tag{7.3}$$

for the ratio of schedule violations. We note that $V = 0$ is required for the correct operation of WTDMA.

We further introduce the starting time delay

$$\phi_{j,b} = s_{j,b} - g_{j,b} \tag{7.4}$$

as the difference between the transmission start time $s_{j,b}$ and the trigger time $g_{j,b}$ of instance $M_{j,b}$.

Finally, the RT $R_{i,a}$ of instance $M_{i,a}$ is computed by adding the starting time delay $\phi_{i,a}$ and the actual message length $L_{i,a}$:

$$R_{i,a} = \phi_{i,a} + L_{i,a}. \tag{7.5}$$

We next provide sufficient conditions for the correct operation of the proposed WTDMA and compute bounds for $\phi_{i,a}$ and $R_{i,a}$.

Theorem 5 *Consider a feasible WTDMA schedule as in Definition 1 with the parameters introduced above. Further, define* $w_{\text{min}} = \min_{M_i \in \mathcal{M}} L_i^{\max}$ *as the minimum window size for the messages in* $\mathcal{M}$ *and* $\Delta_{\max} = \max_{N \in \mathcal{N}} \Delta_N$ *as the maximum software delay of the nodes in* $\mathcal{N}$. *Then,* $V = 0$ *if*

$$w_{\text{min}} > w_{\text{safe}} = \Delta_{\max} + 2 \cdot \theta. \tag{7.6}$$

Hereby, w_{safe} *denotes the safe window size. Furthermore, the starting time delay* $\phi_{j,b}$ *and the RT* $R_{j,b}$ *of any message instance* $M_{j,b}$ *are bounded by*

$$\phi_{j,b} \leq \phi^{\max} = \Delta_{\max} + 2\,\theta \textit{ and } R_{j,b} \leq R_j^{\max} = \phi^{\max} + L_j^{\max}. \tag{7.7}$$

The proof of Theorem 5 is given in Appendix C.1.

That is, (7.6) in Theorem 5 provides a lower bound for the window size depending on the software delay $\Delta_{\max}$ and the maximum clock difference θ. In addition, Theorem

5 allows the computation of the WCRT of each message in $\mathcal{M}$. Hereby, it has to be emphasized that the WCRT for any message M_j in (7.7) only depends on $L_j^{\max}$, θ and $\Delta_{\max}$ and is hence independent of the message priority.

We further note that the window size for each message $M_i \in \mathcal{M}$ depends on its payload size B_i, In particular, the smallest window size is obtained for messages with $B_i = 0$ as $55 \cdot \tau_{\text{bit}}$ (11 bit ID) and $80 \cdot \tau_{\text{bit}}$ (29 bit ID).

In order to assess the practicability of (7.6), we consider realistic values of $\Delta_{\max} = 20\,\mu$s and $\theta = 5\,\mu$s for the software delay and the maximum clock difference. Then, we compare the minimum required safe window size $\Delta_{\max} + 2 \cdot \theta = 30\,\mu$s in (7.6) with the minimum message length w_{min} (11 bit ID) for different bit rates as shown in Fig. 7.2.

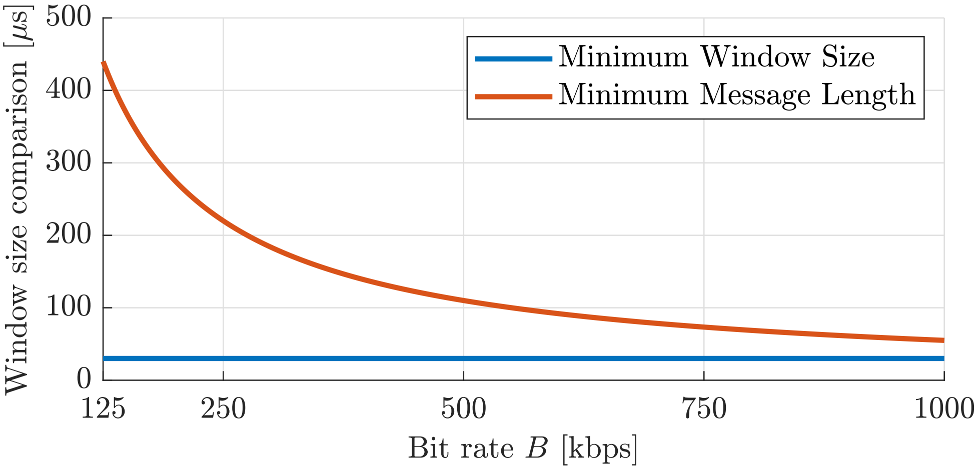

Figure 7.2: Minimum window size and minimum frame length.

It is readily observed that the messages with minimum length are much larger than the minimum required window size even for large bit rates. That is, the proposed WTDMA method is expected to work well in practice.

Remark 13 *The previous discussion focuses on the legacy CAN protocol [4]. We note that an analogous computation of* $w_{\min}$ *can be performed for the recent extensions CAN FD and CAN XL. Specifically, the minimum message length is determined by the duration of the CAN frame fields excluding the payload, which are transmitted at the same maximum bit rate of* 1 *Mbps for all CAN protocols. Hence, the safe window size* w_{safe} *computed for CAN is as well applicable for CAN FD and CAN XL.*

7.4 Advantages of Our Method

We note that there are other methods that provide slotted access to CAN such as TTCAN [26, 100] and the methods in [82, 102]. In this section, we point out various advantages of our method compared to these methods.

First, WTDMA can benefit from the highly accurate CS that is possible with the proposed CS methods in the literature to provide low message RTs, unlike the TDMA approach in [82], where global clocks with an accuracy below 0.5 ms are not considered. Furthermore, WTDMA is not limited to any CS method and can be implemented on MCUs with standard CAN controllers following any of the existing CS methods such as [18]. Thus, it is more advantageous than TTCAN that does not have a global clock for TTCAN Level-1 and depends on its own inherent CS application at the expense of a hardware modification for TTCAN Level-2, as explained in 3.2.2. Although, a software implementation of TTCAN is presented on standard CAN controllers in [102] without requiring the specific TTCAN controllers, the results are not promising for real CAN networks since only a bus load of 12.8% is achieved with a bit rate of $250\,$kbps.

Furthermore, TTCAN protocol has several limitations that are introduced in Section 3.2.2, regarding its Basic Cycle Length, the number of basic cycles in a matrix cycle, the total number of triggers per node and the repetition period of a message trigger. On the contrary, WTDMA provides flexibility in terms of schedule design since it does not have such limitations. That is, feasible WTDMA schedules can be obtained for message sets that are not schedulable on TTCAN. This is confirmed in Section 7.5.3.

Additionally, TTCAN requires a modification of the standard CAN controllers also to provide a timing-related interrupt that notifies the starting of each window [115]. Such a special hardware is not required in WTDMA since it is designed to handle both software delays and clock differences.

In TTCAN, a guard time is applied due to the $TxEnable$ window as explained in Section 3.2.2. Hence, an error frame will prevent the next CAN message if it finishes later than the $TxEnable$ interval of the next window in TTCAN [100]. WTDMA does not apply such a restriction. Specifically, the time windows of different message instances may intersect temporally in our WTDMA. Hence, the next CAN message can start its transmission whenever the error frame finishes since the starting times of the messages are not limited in WTDMA.

7.5 Experimental Evaluation

This section performs a detailed evaluation of the proposed WTDMA method based on experiments. The experimental setup is described in Section 7.5.1 and the considered performance metrics are introduced in Section 7.5.2. Section 7.5.3 to 7.5.9 confirm the correct operation of WTDMA according to the analytical bounds in Theorem 5 in different scenarios including high bit rates, different CS methods and large clock differences, reduced window sizes and the effect of CAN nodes that are not synchronized to the TM.

7.5.1 Experimental Setup and Implementation of WTDMA

The experimental setup consists of 4 FPGA development boards [66] (N1-N4) and 2 MCU boards [65] (N5 and N6). Each board realizes a 32-bit local clock (LC) with an NTU of 250 ns. Clock differences below $\theta = 5\,\mu$s among all nodes are achieved by applying the SW-CS method introduced in [18] with a RM period of $T = 1$ s. The TM node N1 is also responsible for time stamping the reception time instants of all messages on the bus in its own LC and sending the measurements to a PC via an Ethernet connection of 100 Mbps. The contents of the messages are monitored and saved on a PC via a CAN analyzer device. Since the transmission length of a message changes due to bit stuffing, it has to be noted that the actual transmission times in an experiment are deduced by using the recorded contents. Moreover, all CAN messages carry 4-byte generation times in the LC of the transmitter node if there is sufficient space in the payload of the message. Each experiment was conducted until 10^6 CAN messages are transmitted on the bus.

The implementation of WTDMA is realized in software both on the FPGA boards and the MCU boards. The FPGA boards have an ARM Cortex-A9 processor with a CPU clock frequency that is configured as 400 Mhz. The 2 MCU boards include a 32-bit MB91460 series MCU which runs with a core clock of 64 Mhz in the experiments. As described in Section 7.3, the generation of CAN messages is initiated by comparing the LC of the node to the trigger times of the messages via a polling mechanism in software. Thus, variable delays in the generation time of a message can occur relative to the trigger time of the message. Moreover, the generated CAN messages experience some delay before entering the CAN arbitration due to the transfer time to the CAN controller. In our experiments, we verified that the software delays on the FPGA development boards (N1 to N4) and the MCU boards (N5 and N6) are bounded by $\Delta_{N_1 \cdots N_4} = 20\,\mu$s and $\Delta_{N_5} = \Delta_{N_6} = 80\,\mu$s, respectively.

7.5.2 Parameters and Performance Metrics

Using the frame bit length in (2.8) and (2.9) and the message set $\mathcal{M}$, we introduce

$$U_{\max} = \frac{1}{B} \cdot \sum_{i=1}^{m} \frac{f_i^{\max}}{p_i} \quad \text{and} \quad U_{\min} = \frac{1}{B} \cdot \sum_{i=1}^{m} \frac{f_i^{\min}}{p_i}. \tag{7.8}$$

That is, $U_{\max}$ and $U_{\min}$ represent the maximum and minimum bus utilization, which are observed if all messages in $\mathcal{M}$ are transmitted with maximum and minimum bit stuffing, respectively. We further consider a generic experiment, where m_i instances are measured for each message $M_i \in \mathcal{M}$ during a duration T_{exp}. Then, the actual average bus load for the experiment is given by

$$U_{\text{exp}} = \frac{\left(\sum_{i=1}^{m} \left(\sum_{a=1}^{m_i} L_{i,a}\right)\right)}{T_{\text{exp}}}. \tag{7.9}$$

Since the RT depends on the bit stuffing, whereas the starting time delay only depends on the clock differences and software delays, we will focus on the starting time delay $\phi_{i,a}$ to evaluate the performance of our WTDMA.

7.5.3 Message Set

We use the message set in Table 7.1 in order to evaluate the proposed WTDMA method. The message set is constructed such that messages have different payload sizes (B_i in bytes) and periods (p_i in ms). The message periods change between 5 ms and 1000 ms similar to existing message sets such as the SAE and PSA example message sets [86, 116, 117, 51]. The payload sizes are chosen such that there are several short messages with payloads of $0, 1, 2, 3, 4$ Byte since short messages are more challenging for WTDMA according to (7.6). Due to the stuff bits, the actual average bus load U_{exp} can change between $U_{\min} = 83.84\%$ and $U_{\max} = 100\%$ in practice when the bit rate is $B = 250\,\text{kbps}$. The window sizes (w_i) are determined by considering the maximum possible message transmission time for each message. Furthermore, the offset assignment is realized manually in order to obtain a feasible WTDMA schedule. In this context, we want to highlight that the algorithmic offset assignment is not in the scope of this chapter, which focuses on the WTDMA operation. Specifically, the offset assignment is an offline task that is performed before deployment of the CAN system and the assigned offsets do not change during run-time. Computing offset assignments will be the subject of the work in Chapter 8.

We further note that the message set in Table 7.1 is not schedulable on TTCAN. This is due to the fact that $U_{\max} = 100\,\%$ and the RM period is 1 s (M1). On TTCAN, a RM would have to be sent much more frequently, resulting in an infeasible maximum bus utilization above 100%.

7.5.4 TDMA Operation and WCRTs

First, we conduct an experiment with the message set in Table 7.1. The messages are sent by the six nodes in Section 7.5.1 according to the assigned offsets o_i and window sizes w_i given in Table 7.1. Respecting Theorem 5, it holds for the FPGA boards and the MCU boards that $\Delta_{\text{FPGA}} + 2 \cdot \theta = 30\,\mu\text{s}$ and $\Delta_{\text{MCU}} + 2 \cdot \theta = 90\,\mu\text{s}$, respectively. In both cases, the resulting number is smaller than the minimum window length of $w_{\min} = 220\,\mu\text{s}$, which is given by 0-Byte messages. That is, the conditions in Theorem 5 are fulfilled. This result is confirmed by the experimental measurements, where all the messages are transmitted without schedule violations such that $V = 0$. This indicates that WTDMA can be realized with $V = 0$ even for a high average bus load of $U_{exp} = 88\,\%$. This result is confirmed when looking at the measured starting time delays of each message M_i, which remain below $70\,\mu\text{s}$ as can be seen in Fig. 7.3.

Table 7.1: Message and schedule properties.

M_i	o_i	p_i	B_i	w_i	M_i	o_i	p_i	B_i	w_i
M1	0	1000	8	0.54	M21	80.00	200	8	0.54
M2	0.54	1000	8	0.54	M22	120.00	200	8	0.54
M3	1.08	5	0	0.22	M23	160.00	200	8	0.54
M4	1.30	5	0	0.22	M24	15.54	20	8	0.54
M5	1.52	5	0	0.22	M25	10.54	20	8	0.54
M6	1.74	5	0	0.22	M26	20.54	40	8	0.54
M7	1.96	5	0	0.22	M27	40.54	200	8	0.54
M8	2.18	5	1	0.26	M28	80.54	200	8	0.54
M9	2.44	5	1	0.26	M29	120.54	200	8	0.54
M10	2.70	5	1	0.26	M30	160.54	200	8	0.54
M11	2.96	5	2	0.30	M31	200.00	1000	8	0.54
M12	3.26	5	2	0.30	M32	400.00	1000	8	0.54
M13	3.56	5	3	0.34	M33	600.00	1000	8	0.54
M14	3.90	5	3	0.34	M34	800.00	1000	8	0.54
M15	4.24	5	4	0.38	M35	200.54	1000	8	0.54
M16	4.62	5	4	0.38	M36	400.54	1000	8	0.54
M17	15.00	20	8	0.54	M37	600.54	1000	8	0.54
M18	10.00	20	8	0.54	M38	800.54	1000	8	0.54
M19	20.00	40	8	0.54	M39	5.00	20	8	0.54
M20	40.00	200	8	0.54	M40	5.54	20	8	0.54

o_i, p_i and w_i are given in ms. $U_{\max}$ = 100.0% when $B = 250$ kbps

This also validates (7.7) since $w_{\text{min}} > w_{\text{safe}}$. It can further be seen that the starting time delays of M19, M20, M21, M22, M39 and M40 are larger compared to the other messages. The reason is that these messages are transmitted by the MCU boards N5 and N6 which encounter larger internal software delays as indicated in Section 7.5.1.

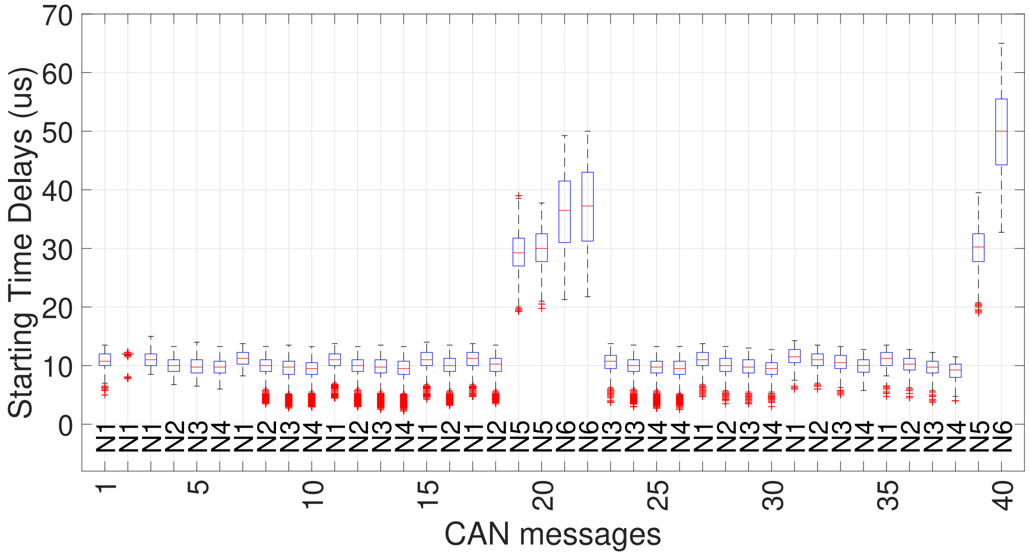

Figure 7.3: Starting time delays for a bit rate of $B = 250\,\text{kbps}$.

We repeat the same experiment with an increased bit rate of $B = 1\,\text{Mbps}$. Hereby, we adapt the message set in Table 7.1 by dividing the parameter values o_i, p_i and w_i by 4 in order to achieve the same bus utilization as in the first experiment. In this case, $w_{\text{min}} = 55\,\mu$s for CAN messages with zero payload ($M_3 - M_7$). Since the software delay of the MCU boards (N5 and N6) is too large for this minimum window size, we distribute all messages to the 4 FPGA boards. The measured starting time delays of the messages are shown in Fig. 7.4. It is seen that all starting time delays remain below $20\,\mu$s, which implies that the WTDMA operation is again successfully carried out for the bit rate of $1\,\text{Mbps}$. Hereby, the average bus load U_{exp} is measured very high as 89% during the experiment.

7.5.5 Evaluation of Computed Bounds

In this section, we evaluate the validity of the bound computed in Theorem 5. To this end, we conduct experiments that modify w_{safe} by artificially adjusting the maximum

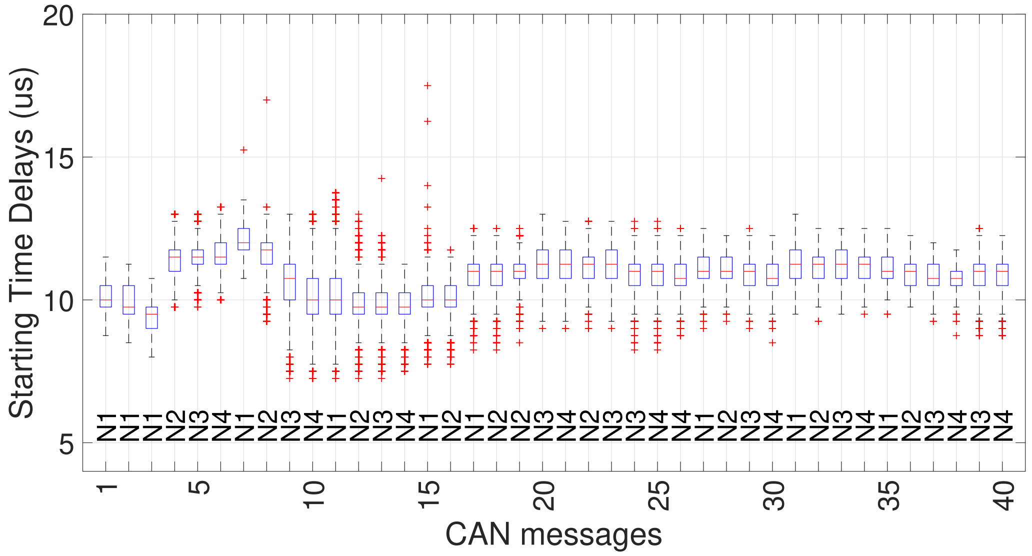

Figure 7.4: Starting time delays for a bit rate of $B = 1\,\text{Mbps}$.

clock difference θ. This is achieved by adding/subtracting random values in a specified range from the synchronized clocks of CAN nodes. We use the message set in Table 7.1 and a bit rate of $B = 250\,\text{kbps}$. Considering that $w_{\min} = 220\,\mu\text{s}$, we run experiments for the values $\theta_1 = 5\,\mu\text{s}$ (no modification), $\theta_2 = 100\,\mu\text{s}$, $\theta_3 = 120\,\mu\text{s}$, $\theta_4 = 130\,\mu\text{s}$ and $\theta_5 = 200\,\mu\text{s}$. That is, schedule violations are expected for θ_3, θ_4 and θ_5 according to (7.6). Table 7.2 shows the resulting ratio of schedule violations.

Table 7.2: Observed ratios of schedule violations.

	θ_1	θ_2	θ_3	θ_4	θ_5
V	0	0	0.0025	0.0031	0.0083

As expected, there are no schedule violations for the case of θ_1 and θ_2, whereas an increasing number of schedule violations is observed with larger clock differences. In order to further illustrate the concept of schedule violations, we select the messages M3 to M9 in Table 7.1, which are transmitted successively according to the WTDMA schedule. According to Theorem 5, we expect that the groups of the starting times of these messages do not overlap (for example, the starting times of M3 should always be before the starting times of M4 if there are no schedule violations). The distribution of the starting times of the selected messages is shown in Fig. 7.5. It can be seen that the message transmission order is correct even for clock differences as high as $100\,\mu$s since there is no intersection of the groups of starting times. Differently, schedule violations are observed for θ_3, θ_4 and θ_5. In particular, it can be seen that the starting times of the different messages are separated into two blocks. This indicates that the successor message (for example M4) overtakes the predecessor message (for example M3), which is in line with the computed bound in (7.6). It is interesting to note that no overtakes between the messages M9 and M8 are observed when $\theta_3 = 120\,\mu$s. This is due to the fact that the payload of M8 is $B_8 = 8\,$Byte and hence its window size $w_8 = 260\,\mu$s is larger than $w_{\text{safe}} = 245\,\mu$s. Together, this experiment confirms the correctness of the (7.6).

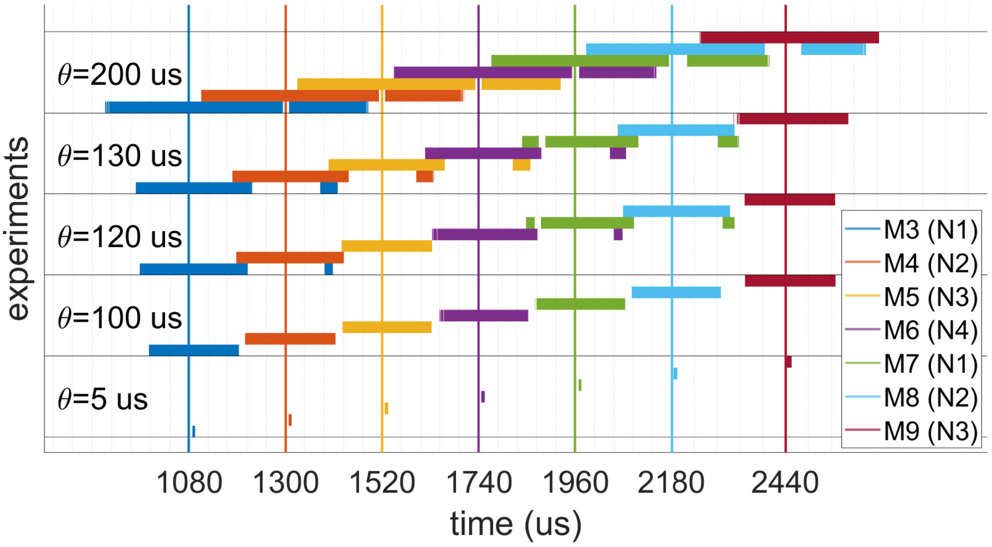

Figure 7.5: The distribution of starting times of messages.

7.5.6 Usage of a Reduced Window Size

The conventional scheduling methods consider the message transmission time as the maximum message length which is only possible when the maximum number of stuff bits are added. However, maximum stuff bits are highly unlikely in real CAN networks [118]. Thus, we also realized an experiment, where the window sizes are computed by considering the message transmission time statistics in our previous experiments. The message set in Table 7.1 is modified by changing the periods and payload lengths of M17, M24, M39 and M40 as shown in Table 7.3. In this case, the expected bus utilization is in the order of 100 % but can theoretically reach $U_{\max} = 108.88\%$ if all messages experience the maximum stuff bits. Additionally, a new WTDMA schedule is designed such that new offsets and window sizes for all messages are determined by considering the average message lengths. Hereby, the minimum window size is shortened to 200 μs in this experiment.

Table 7.3: Properties of the modified messages.

M_i	p_i	B_i	M_i	p_i	B_i
M17	10	8	M24	10	8
M39	5	0	M40	5	0

Although the average bus load is measured as 96.2%, the message transmission order is correct. The starting time delays can be seen in Fig. 7.6. The messages M3-M16 are sent successively also in this experiment. Besides, M39 is sent right after M16 according to the WTDMA schedule. If the actual transmission time of a message is greater than its window size, the starting time of the next message will be blocked since CAN is a non-preemptive bus. In case, transmission times of the successive messages exceed their window sizes, the sum of excess times creates a cumulative effect on the following messages. Thus, the peak values of the starting times for M3-M16 and M39 are in an increasing order in Fig. 7.6. Even though delays up to 85 μs are observed for M39, it does not result in any violation in the transmission orders in our WTDMA. To understand how the WTDMA operation functions successfully in this case, it has to be remembered that the effect of the excess time is compensated when the actual transmission length of a message is shorter than its window size which is also equally possible with longer messages. To sum up, our WTDMA approach works well even for bus loads that are above 95% and makes it possible to have message transmissions with very low starting times and hence also RTs.

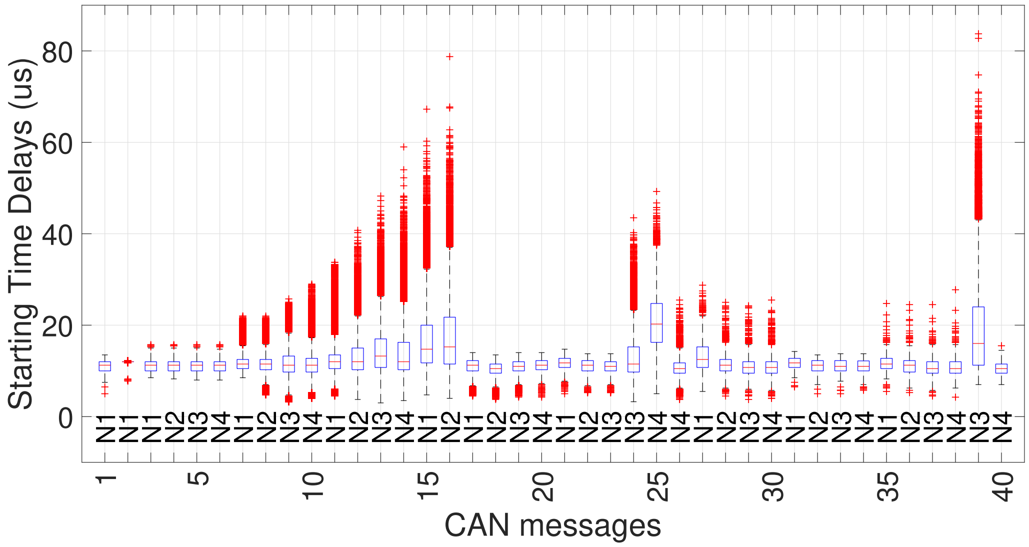

Figure 7.6: The starting time delays of CAN messages - 250 kbps with smaller window sizes.

7.5.7 Independence of the CS Method

As indicated in Section 7.2, the proposed WTDMA protocol is independent of the CS method. Thus, any CS algorithm in the literature can be used together with WTDMA. In order to demonstrate this fact, we realized an experiment by using the message set in Table 7.1, where N1 as TM sends RMs as described in [27]. N2, N3 and N4 apply the AUTOSAR CS method in [27] as slave nodes. N5 and N6 apply the controller based CS method in [18]. It is verified that the message transmission orders are correct as expected during the experiment where CAN nodes apply different CS methods since the safe window size w_{safe} with the clock differences of the applied CS methods remain below the smallest window size.

7.5.8 CAN Bus with Fault Injection

In case of a bit error on a CAN message, a maximum of 20 extra bits as an error frame can be transmitted by the CAN nodes detecting the bit error on the bus before the IFS starts prior to the next CAN message. It has to be noted that bit errors occurring earlier than the last 20 bits of the CAN message do not affect the next CAN message when using WTDMA since the error frame ends before the expected finishing time of the erroneous CAN message. The worst case happens for a bit error in the last bit of the CAN message. Thus, to formally guarantee a successful WTDMA operation under bit errors at the end of each CAN message, Definition 1 could be extended by including a possible error frame for each CAN message such that $L_i^{max} + 20 \cdot \tau_{bit} \leq w_i$ is required.

However, since CAN messages do not experience their maximum transmission lengths frequently in practice, the suggested extension would lead to an inefficient bandwidth use. Instead, it is possible to use the free spaces due to messages without maximum bit stuffing to accommodate the error frames in WTDMA since the starting times of the CAN messages are not restricted to a tight interval unlike TTCAN. That is, even if a WTDMA schedule is constructed by considering the original message lengths, the error frames will be handled successfully in WTDMA unless bit errors occur during the last bits of all CAN messages on the bus. However, in this case, the CAN bus it not operational anyway.

In order to demonstrate the error tolerance of WTDMA, we repeated the experiment in Section 7.5.4 by injecting errors from the CAN analyzer device with a bit error rate (BER) of 10^{-3}. Although the bit error generation in the experiment is much more challenging compared to practical values in the literature, where the BER for CAN is in the order of 10^{-10} in benign environments and 10^{-6} in aggressive environments [119], no violation ($V = 0$) was observed in the WTDMA schedule. To sum up, it is verified that WTDMA can handle error frames with a high BER without any performance loss even at a high measured bus load of 90$\%$.

7.5.9 CAN Nodes with Non-Sync Messages

We next consider the case where some messages are not transmitted according to the WTDMA schedule. This case is for example possible when one of the nodes on the CAN bus does not have an synchronized clock and hence does not realize WTDMA. In order to evaluate this scenario, we repeat the experiment in Section 7.5.4 with a change such that N6 does not follow a SW-CS method. Thus, the non-sync messages M21, M22 and M40 that are sent by N6 do not fulfill (7.6) and hence violate the required rules of the WTDMA protocol. As a consequence, it is experimentally observed that the order of the synchronized messages changes due to the non-sync messages. That is, formally, the WTDMA operation fails since the order of the messages can not be strictly followed unless all nodes apply SW-CS.

When the RTs are evaluated, the most affected message is M16 whose RT is presented in Fig. 7.7 together with M17, M18 and M15 as examples. Hereby, the RT of a message can not be shorter than its actual transmission time which can change from 316 μs to 380 μs for M15 and M16, and from 444 μs to 540 μs for M17 and M18. By considering the minimum possible RTs, it can be seen in Fig. 7.7 that the WTDMA operation returns to its correct operation even after the WCRTs are experienced. That is, the saw-tooth shape of the RTs shows us that the effect of non-sync messages is not permanent. Even though the order of the messages can not always be preserved throughout the experiment, the benefits of applying our WTDMA are still observable. To sum up, our WTDMA protocol can be followed also together with non-sync nodes at the expense of a reasonable performance loss in terms of RTs.

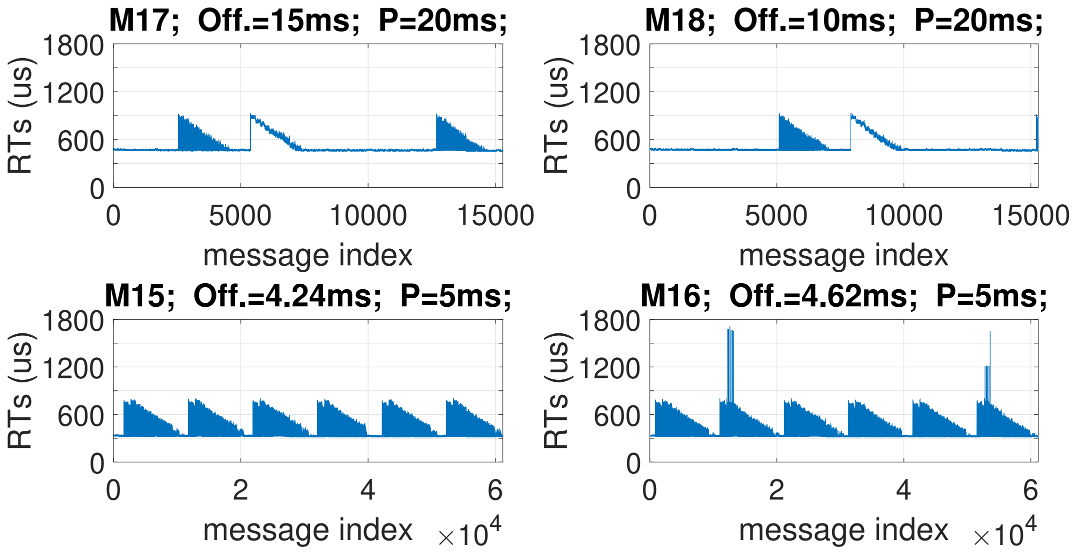

Figure 7.7: Response times of messages.

7.6 Discussion and Conclusion

Providing deterministic medium access on the controller area network (CAN) both enables a more efficient bandwidth usage and improves the real-time capabilities of CAN such as message latencies. This chapter introduces WTDMA in order to provide slotted medium access on CAN. Thus, the traffic shaping layer that is another significant component of CANDS framework is realized according to the proposed WTDMA model.

As an important feature, WTDMA makes use of the medium access CSMA/CR and the non-preemptive message transmission of CAN by allowing a certain degree of overlap of time windows. As a result, WTDMA can be realized with a moderate accuracy of the synchronized clocks of different CAN nodes and can hence be implemented in software without any modifications to the CAN standard. The chapter formally shows under which conditions WTDMA operates correctly and provides an experimental evaluation to validate the formal results in practice. The experiments confirm that high bus loads above 90% can be achieved at message latencies that are only determined by the message transmission time. In addition, WTDMA is shown to be tolerant to bit errors and remains operational even if a limited number of messages that do not follow the WTDMA schedule are transmitted on the bus.

CHAPTER 8

WTDMA MESSAGE SCHEDULING

A WTDMA schedule is defined by the message offsets $\mathcal{O}$ and the message window sizes $\mathcal{W}$, as explained in the WTDMA model in Chapter 7. Determining the message offsets and window sizes are realized offline and is called as WTDMA message scheduling. During the message scheduling, the offsets and window sizes should be arranged such that all messages have their own time slots. That is, any message window cannot be shared completely or partially between two different CAN messages.

Although there are existing CAN offset assignment works in the literature [35, 83, 92], they are not applicable to WTDMA scheduling problem since they are developed for the case where there is not a global clock among the nodes. They realize offset assignments for messages that are sent from a single node, by ignoring the other messages. On the contrary, WTDMA model depends on a global clock where all messages are transmitted by respecting the global clock on the network. Thus, new WTDMA scheduling algorithms that consider all messages on the bus are required.

In this chapter, we propose to realize the WTDMA schedule design in two steps. Specifically, one HP is divided into mini time slots with equal lengths. The placement of the CAN messages into those mini time slots is realized as the first step. Afterwards, the ultimate offset assignments are determined in the second step. As one of the contributions in this chapter, the first and second steps are formulated as ILP optimization problems which can be solved by standard ILP solvers such as CPLEX [120]. Hereby, it has to be noted that the ILP formulations in [121, 122] which are introduced for different problems gives inspiration while formulating the WTDMA scheduling as ILP problems. Additional to the ILP-based methods, heuristic approaches are also developed for the first and second steps, separately.

The proposed methods are evaluated with several randomly generated message sets that are in line with the real automotive applications. It is seen that the run-time of the first step ILP (ILP-S1) method can be as high as 30 minutes when bus load is close to 100%. Moreover, the second step ILP (ILP-S2) can last up to $4\,\mathrm{s}$. The achieved run-times with ILP-based methods are practicable since a schedule design is not repeated very often. Regarding the heuristic methods, the second step heuristic (H-S2) provides a close performance to ILP-S2 in terms of run-times and ability to solve the problem. The first step heuristic (H-S1) is not as successful as ILP-S1 in finding a solution when the bus load is close to 100%. However, the run-time of H-S1 is independent from the bus load and is always measured as less than 1 s. Thus, H-S1 can be applied before ILP-S1 in order to find a solution throughout the first step.

8.1 Background

The length of one hyper-period (HP) is defined as $H = \mathrm{lcm}(p_1, \dots, p_m)$ in Chapter 7. It is assumed that one HP duration is divided into the mini time slots which have equal lengths. The duration of one mini time slot L_s is defined as the GCD of the periods of the messages $M_i \in \mathcal{M}$ such that $L_s = gcd(p_1, ..., p_m)$. Therefore, there will be $S = \frac{H}{L_s}$ slots in one HP and the slot indexes are numbered from 0 to $(S - 1)$.

While the placement of the messages into the mini time slots, s_i represents the index of the earliest slot where the message M_i appears and the repetition period q_i is given by $q_i = \frac{p_i}{L_s}$. Thus, s_i can take the values between 0 and $q_i - 1$ since mini time slot offset assignment s_i should be smaller than the mini time slot repetition period q_i. That is, the CAN message M_i exists in all mini time slots $s_i + k \cdot q_i$ for $k = 0, ..., \frac{S-1}{q_i}$

The offset assignment for WTDMA scheduling is realized in two steps in this chapter. In the first step, the messages are assigned into the mini time slots such that the total lengths of the messages in a slot should not exceed the length of the slot L_s. Then, the final message offset assignment o_i for message M_i is decided by considering the determined slot assignments such that the window intervals of different messages do not overlap as required in a feasible WTDMA schedule.

8.2 Integer Linear Programming for WTDMA Scheduling

8.2.1 ILP for the First Step - Mini Time Slot Assignment

The first step (mini time slot assignment) problem is formulated as ILP such that

1. The binary decision variables $(x_{i,0}, ...x_{i,q_i-1})$ defines the slot assignment for each message $M_i \in \mathcal{M}$. That is, the slot index s_i of M_i is given as $s_i = x_{i,0} \cdot 0 + x_{i,1} \cdot 1 + ... + x_{i,q_i-1} \cdot (q_i - 1)$ by the binary decision variables of M_i .
2. In order to have a balanced distribution among the mini time slot usages, it is required to keep the maximum load of the mini time slots T_{ML} as small as possible.
3. Only one of the variables $x_{i,0}, ...x_{i,q_i-1}$ can be 1 whereas the other variables must be zero for every message $M_i \in \mathcal{M}$.
4. The total length of the messages contained in any mini time slot must not exceed the length of the mini time slot.

The decision vector x consists of all decision variables for all messages $\forall i \in \{1, ..., m\}$ together with T_{ML} as written below:

$$x = \begin{bmatrix} x_{1,0} \\ \vdots \\ x_{1,q_1-1} \\ \vdots \\ x_{m,0} \\ \vdots \\ x_{m,q_m-1} \\ T_{ML} \end{bmatrix}$$

It has to be noted that the decision variables $x_{i,j}$ for $i = \{1, ..., m\}$ and $j = \{0, ..., q_i - 1\}$ in x matrix are binary. However, T_{ML} can take integer values that shows the amount of the highest usage among the mini time slots.

Even though the constraint 2) is not mandatory as long as constraint 4) is satisfied, it is preferred to have a balanced distribution among the mini time slots. Therefore, the objective function which is written as $J = \min_x T_{ML}$ is preferred to be minimized.

The constraint 3) is written for an arbitrary message M_i as:

$$\sum_{j=0}^{q_i-1} x_{i,j} = 1. \tag{8.1}$$

and it has to be true for each $M_i \in \mathcal{M}$ with $\forall i \in \{1, ...m\}$.

Furthermore, the constraint 4) has to be true for all mini time slots from 0 to $S-1$ and it is formulated as :

$$\sum_{i=1}^{m} (L_i^{max} \cdot x_{i,(k \mod q_i)}) \leq L_s, 0 \leq k \leq S-1. \tag{8.2}$$

The equation 8.2 guarantees that total message length in any mini time slot $(0, 1, ..., S-1)$ does not exceed the mini time slot length L_s. Hereby, a CAN message M_i that is assigned to slot j (implying that $s_i = j$ and hence $x_{i,j} = 1$) takes place in the mini time slot with slot index k whenever $k \mod q_i = j$ is true.

By considering the provided explanations, ILP problem can be formulated as compatible to the following standard form:

$$\min_x f \cdot x. \tag{8.3}$$

$$A_{eq} \cdot x = b_{eq}. \tag{8.4}$$

$$A \cdot x \leq b. \tag{8.5}$$

in order to be able to solve it with standard ILP solvers such as CPLEX [120].

We next continue with an example CAN message set in Table 8.1 to illustrate the ILP formulation for the first step. Accordingly, the mini time slot length L_s equals to 1 ms for this message set and there are 4 mini time slots during one HP whose duration is 4 ms.

The x matrix is computed for this example as :

$$x^T = \begin{bmatrix} x_{1,0} & x_{2,0} & x_{2,1} & x_{3,0} & x_{3,1} & x_{4,0} & x_{4,1} & x_{4,2} & x_{4,3} & x_{5,0} & x_{5,1} & x_{5,2} & x_{5,3} & T_{ML} \end{bmatrix}$$

Table 8.1: Example of CAN message set.

M_i	p_i [μs]	$L_i^{\max}$ [μs]
M_1	1000	100
M_2	2000	120
M_3	2000	140
M_4	4000	160
M_5	4000	180

Moreover, the f matrix is constructed as:

$$f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

in order to minimize the highest mini time slot usage T_{ML}.

Furthermore, A_{eq} and b_{eq} matrices are constructed for the given example as follows:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \cdot x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

A and b matrices for the inequality equation are as follows:

$$\begin{bmatrix} 100 & 120 & 0 & 140 & 0 & 160 & 0 & 0 & 0 & 180 & 0 & 0 & 0 & -1 \\ 100 & 0 & 120 & 0 & 140 & 0 & 160 & 0 & 0 & 0 & 180 & 0 & 0 & -1 \\ 100 & 120 & 0 & 140 & 0 & 0 & 0 & 160 & 0 & 0 & 0 & 180 & 0 & -1 \\ 100 & 0 & 120 & 0 & 140 & 0 & 0 & 0 & 160 & 0 & 0 & 0 & 180 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\cdot x \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1000 \end{bmatrix}$$

The constructed ILP problem is solved by CPLEX ILP solver [120] and the x matrix is found as follows:

$$\begin{aligned} x^T &= \begin{bmatrix} x_{1,0} & x_{2,0} & x_{2,1} & x_{3,0} & x_{3,1} & x_{4,0} & x_{4,1} & x_{4,2} & x_{4,3} & x_{5,0} & x_{5,1} & x_{5,2} & x_{5,3} & T_{ML} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 360 \end{bmatrix} \end{aligned}$$

According to the definition of x matrix, the maximum mini time slot usage is seen as 360 μs. Moreover, the mini time slot offset assignments s_i values are seen as $s_1 = 0$, $s_2 = 0$, $s_3 = 0$, $s_4 = 1$ and $s_5 = 3$ as shown in Table 8.2.

Table 8.2: The s_i values found by ILP.

s_1=	$x_{1,0} \cdot 0$	= 0
s_2=	$x_{2,0} \cdot 0 + x_{2,1} \cdot 1$	= 0
s_3=	$x_{3,0} \cdot 0 + x_{3,1} \cdot 1$	= 0
s_4=	$x_{4,0} \cdot 0 + x_{4,1} \cdot 1 + x_{4,2} \cdot 2 + x_{4,3} \cdot 3$	= 1
s_5=	$x_{5,0} \cdot 0 + x_{5,1} \cdot 1 + x_{5,2} \cdot 2 + x_{5,3} \cdot 3$	= 3

The distribution of the messages in the mini time slots for one HP is represented in Fig. 8.1. That is, s_1 is 0 and M_1 are sent in the mini time slots 0, 1, 2 and 3 since its repetition period q_1 is 1. s_2 and s_3 is also 0 and M_2 and M_3 are sent in the mini time slots 0 and 2 since their repetitions q_2 and q_3 are 2. Moreover, s_4 is 1 and M_4 is sent in the mini time slot 1. s_5 is 3 and M_5 is sent in the mini time slot 3. Since their repetitions q_4 and q_5 are 4, they are transmitted only one time in one HP whose length is equal to 4 mini time slots.

8.2.2 ILP for the Second Step - Ultimate Offset Assignment

It is expected that all CAN messages are placed into the mini time slots successfully at the end of the first step. It is possible that an instance of CAN message M_i may be placed together with a group of CAN messages in a mini time slot, but a different group of CAN messages may exist in another mini time slot together with another instance of M_i, when M_i is transmitted more than one time in one HP. The aim of the second step is to find the ultimate offset assignments such that the window intervals construct a feasible schedule as defined in Definition 1. That is, the ultimate offset

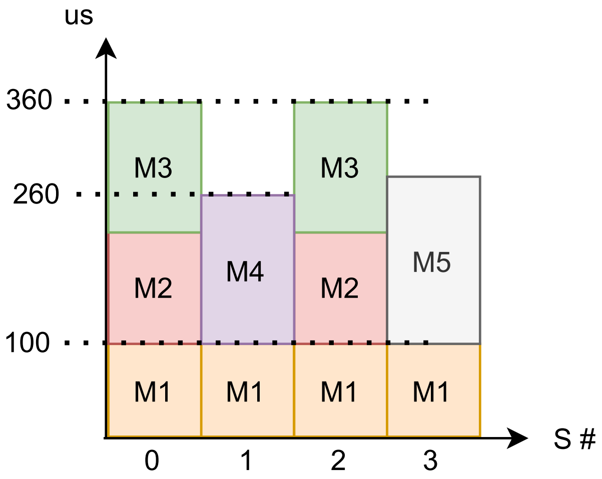

Figure 8.1: Example of mini time slot assignment.

table $\mathcal{O} = o_1, ..., o_m$ is determined by respecting to the mini time slot assignments $s_1, ..., s_m$ that is the output of the first step. Specifically, the second step determines the offset table $\overline{\mathcal{O}} = \overline{o}_1, ..., \overline{o}_m$ as the offsets of the messages within the mini time slot. Thus, any offset value $\overline{o}_i$ can not be bigger than the mini time slot length L_s, which can be expressed as $\overline{o}_i \leq L_s, \forall i \in 1, ..., m$. In order to determine the the offset table $\overline{\mathcal{O}} = \overline{o}_1, ..., \overline{o}_m$, an ILP formulation is developed to solve the problem as compatible to the standard ILP equations in 8.3, 8.4 and 8.5. Together with the offset table $\overline{\mathcal{O}}$ values, the ultimate offset table $\mathcal{O}$ values can be computed as follows:

$$o_i = \overline{o}_i + s_i \cdot L_s, \forall i \in 1, ..., m. \tag{8.6}$$

Regarding the second step ILP, the binary decision variables are chosen as below:

$$x^T = \begin{bmatrix} \overline{o}_1 & \cdots & \overline{o}_m & u_{1,2} & \cdots & u_{1,m} & u_{2,1} & \cdots & u_{2,m} & \cdots & \cdots & u_{m,i} & \cdots & u_{m,m-1} \end{bmatrix}$$

where $u_{i,j}$ is a binary variable which defines the positions of the CAN messages M_i and M_j according to each other. That is, if M_i is sent earlier than M_j in the mini time slot, $u_{i,j}$ takes the value of 0. However, $u_{i,j}$ takes the value of 1 when M_i is sent later than M_j. Hereby, $u_{i,j}$ is defined for $i = \{1, ..., m\}$, $j = \{1, ..., m\}$ and $i \neq j$. Furthermore, it has to be noted that $u_{i,j}$ and $u_{j,i}$ have always the different values (0 or 1) according to its definition.

Moreover, the co-existence of M_i and M_j CAN messages in the same mini time slot together is presented with another variable $c_{i,j}$. According to the mini time slot assignment after the first step, if M_i and M_j are placed in any mini time slot together, $c_{i,j}$ takes the value of 0. On the contrary, $c_{i,j}$ takes the value of 1, if M_i and M_j do not take place together in any of the mini time slots. The values of $c_{i,j}$ are determined directly according to the mini time slot schedule after the first step. Hereby, $c_{i,j}$ is defined $\forall i, j \in \{1, ..., m\}$ and $c_{i,j} = 0$ when $i = j$ as compatible to its definition. Lastly, it has to be noted that the value of the binary decision variable $u_{i,j}$ is not important and take 0 or 1 when M_i and M_j are not located in the same mini time slot ($c_{i,j} = 1$).

We next continue with the first constraint that is below:

$$\overline{o}_i + L_i^{max} \leq L_s, \forall i \in \{1, ...m\} \tag{8.7}$$

According to the constraint 8.7 , all CAN message $M_i \in \mathcal{M}$ must complete its transmission before the end of the mini time slot. That is, in the second step it is not allowed for a CAN message to exceed the bounds of the mini time slot while trying to find an offset value $\overline{o}_i$. In this way, the output of the first step is respected, and the content of the different mini time slots are kept isolated from each other during the second step.

The second constraint which avoids the overlap between different CAN message windows within a mini time slot is provided as follows:

$$\overline{o}_i + L_i^{max} \leq \overline{o}_j + u_{i,j} \cdot L_s + c_{i,j} \cdot L_s, \forall i, j \in \{1, ...m\}, i \neq j. \tag{8.8}$$

Specifically, the equation 8.8 guarantees that the transmission of M_i must be completed earlier than M_j in case of that M_i and M_j are placed in the same mini time slot and M_i is placed earlier than M_j. That is, the equation 8.8 is always true in case of that M_i is scheduled later than M_j and hence $u_{i,j} = 1$, due to the first constraint defined by the equation 8.7. Similarly, when M_i and M_j are not placed in the same mini time slot, $c_{i,j} = 1$ and the equation 8.8 is always true due to the first constraint in the equation 8.7.

Lastly, the third constraint which comes from the definition of $u_{i,j}$ variables is presented as follows:

$$u_{i,j} + u_{j,i} = 1, \forall i, j \in \{1, ...m\}, i \neq j. \tag{8.9}$$

According to the constraint (8.9), only one of the $u_{i,j}$ and $u_{j,i}$ binary variables can be 1.

The second step ILP approach is further illustrated with an example CAN message set in Table 8.3.

Table 8.3: Example of CAN message set.

M_i	p_i [μs]	$L_i^{\max}$ [μs]
M_1	3000	250
M_2	4000	500
M_3	6000	250

The length of one HP equals to 12 ms as $lcm(p_1, p_2, p_3)$ and the mini time slot length L_s is 1 ms as $gcd(p_1, p_2, p_3)$.

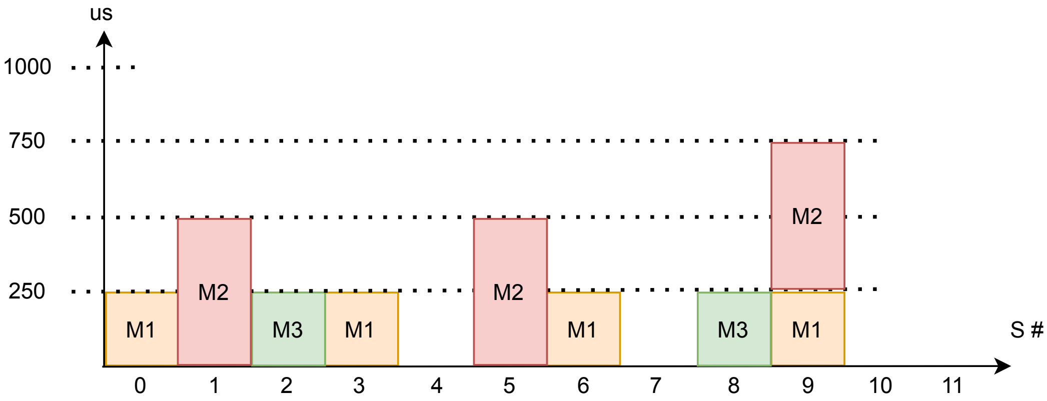

Figure 8.2: Example of mini time slot schedule.

It is assumed that the mini time slot assignment is realized by the first step ILP and the schedule in Fig. 8.2 is found. According to the schedule, mini time slot assignments are seen as $s_1 = 0$, $s_2 = 1$, and $s_3 = 2$. In addition, $c_{i,j}$ values are as follows:

$$\begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,1} & c_{3,2} & c_{3,3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

In the mini time slot 9, M_1 and M_2 messages are placed together. Thus, $c_{1,2} = c_{2,1} = 0$. Furthermore, $c_{1,1} = c_{2,2} = c_{3,3} = 0$ by definition of $c_{i,j}$ since $i = j$.

The ILP decision vector x is determined as:

$$x^T = \begin{bmatrix} \overline{o}_1 & \overline{o}_2 & \overline{o}_3 & u_{1,2} & u_{1,3} & u_{2,1} & u_{2,3} & u_{3,1} & u_{3,3} \end{bmatrix}$$

The constraints in 8.7 and 8.8 are presented with inequality matrices A and b by rewriting the constraints as follows:

$$\overline{o}_i \leq L_s - L_i^{max}, \forall i \in \{1, ...m\} \tag{8.10}$$

$$\overline{o}_i - \overline{o}_j - (u_{i,j} \cdot L_s) \leq (c_{i,j} \cdot L_s) - L_i^{max}, \forall i, j \in \{1, ...m\}, i \neq j. \tag{8.11}$$

Specifically, the matrices A and b are constructed for the given example as follows:

$$\begin{bmatrix} 1 & -1 & 0 & -1000 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1000 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & -1000 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & -1000 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & -1000 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & -1000 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot x \leq \begin{bmatrix} -250 \\ 750 \\ -500 \\ 500 \\ 750 \\ 750 \\ 750 \\ 500 \\ 750 \end{bmatrix}$$

Furthermore, the equality matrices A_{eq} and b_{eq} are constructed as:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \cdot x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

In the second step ILP formulation, it is not required to minimize any of the variables in x. Thus, f matrix is chosen as 0. The given example is solved by CPLEX ILP solver such that :

$$x = \begin{bmatrix} \overline{o}_1 \\ \overline{o}_2 \\ \overline{o}_3 \\ u_{1,2} \\ u_{1,3} \\ u_{2,1} \\ u_{2,3} \\ u_{3,1} \\ u_{3,3} \end{bmatrix} = \begin{bmatrix} 250 \\ 500 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Hereby, there may be multiple solutions that satisfy the requirements. However, the offset assignments for CAN messages are found by ILP solver as $\overline{o}_1 = 250\,\mu$s, $\overline{o}_2 = 500\,\mu$s and $\overline{o}_3 = 0\,\mu$s in our example. Together with s_i and determined $\overline{o}_i$ values, the ultimate offset values are found as $o_1 = 250\,\mu$s, $o_2 = 1500\,\mu$s and $o_3 = 2000\,\mu$s. By considering the message length and periods in the Table 8.3, it can be seen that the CAN messages are transmitted without any collision with the assigned offset o_i values after the second step ILP.

8.3 The Heuristic Method for WTDMA Scheduling

8.3.1 The Heuristic Method for the First Step - Mini Time Slot Assignment

In order to place the messages into the mini time slots, heuristic approaches may also be used as an alternative to ILP-based methods. The messages should be placed into the mini time slots whose lengths are L_s with respect to the same constraints that are already explained for the ILP for the first step (8.2.1). It has to be remembered that the total usage of a mini time slot should not be more than the length of the mini time slot L_s and the first instance of a message M_i should be placed in a mini time slot with an index s_i that is smaller than its repetition period q_i.

The heuristic method for the first step is presented in Algorithm 9. As the initialization, the values for the mini time slot assignments s_i are empty. The messages are ordered in line 4 and the mini time slot placements are realized in the rest of the algorithm according to this order. Therefore, the order of messages after line 4 determines the performance of the heuristic algorithm. Although there are several ordering options, we choose to order messages according to their periods. Specifically, the message with the smallest period will be placed on top of the queue Q_{mes} such that it will be processed first for the mini time slot assignment. When there are multiple messages with the same period, the ordering is realized randomly. After the messages are ordered, the mini time slot assignments are realized for each message until all messages are processed (line 6-8). In line 7, the possible mini time slot assignments ($0 \leq s_i \leq q_i - 1$) are compared and the option which results in the lowest maximum mini time slot usage T_{ML} is preferred. When there are multiple options, any of s_i options can be chosen randomly. After deciding its s_i value, the present message is removed from the queue Q_{mes} so that the next message can be processed. In the end, the feasibility of the mini time slot assignments $\mathcal{S}$ are checked (line 9-13).

Algorithm 9 WTDMA Scheduling Algo - Heuristic

```
1:  Input: $\mathcal{M}$
2:  Output: $\mathcal{S} = s_1, ..., s_m$
3:  Init: $s_i = \emptyset$ for $1 \leq i \leq m$; $H = gcd(p_1, ..., p_m)$ ; $L_s = lcm(p_1, ..., p_m)$ ;
        $T_{ML} = 0$ ; $Q_{mes} = \emptyset$.
4:  Fulfill the message list as a queue $Q_{mes}$ by ordering messages in $\mathcal{M}$ according to
        their periods.
5:  Remove $M_i$ from the queue $Q_{mes}$.
6:  while $Q_{mes}$ is not empty do
7:      $s_i$ assignment for the present message $M_i$.
8:  end while
9:  if $T_{ML} > L_s$ then
10:     No feasible solution at the end of the $1^{st}$ stage!
11: else
12:     The $1^{st}$ stage is successful.
13: end if
```

When the algorithm fails to find a feasible solution, the same algorithm can be followed again with a change that the message order in line 4 is realized according to another criteria such as the message length. Changing the order of the messages directly affects the output of the algorithm and may help to find a feasible solution.

8.3.2 The Heuristic Method for the Second Step - Ultimate Offset Assignment

In order to find the ultimate offset assignments $\mathcal{O} = o_1, ..., o_m$ for CAN messages, an heuristic in 10 is developed. In the algorithm, a queue Q_{mes} which follows the First In First Out (FIFO) principle and a stack S_{proc} that follows the Last In First Out (LIFO) principle are used. Q_{mes} consists of the messages whose offset values are not assigned yet. However, S_{proc} consists of the messages whose process is complete and have assigned offset values. That is, the union of the contents of Q_{mes} and S_{proc} gives the CAN message set $\mathcal{M}$.

Before starting to process the messages, the queue Q_{mes} is fulfilled in line 4 that determines the order of the messages for the rest of the algorithm. Here, we choose to order the messages according to their periods in the same way with the first step heuristic. Moreover, the algorithm in 10 inputs mini time slot assignments S and constructs dependency lists $W_{dep,i}$ for each message. $W_{dep,i}$ consists of the messages that share the same mini time slot with any instance of M_i. Then, the algorithm starts processing messages one by one (line 6).

The present message is read in line 7. Then, in line 8, an imaginary timeline is constructed where the messages in the dependency list of the present message are transmitted at their relative offsets $mod(o_i, L_s)$. The imaginary timeline goes from 0 to L_s. The messages in the dependency list that have not been processed yet are not be included in the imaginary timeline. Afterwards, it is checked if there is at least one empty interval in the imaginary timeline which is longer than the length of the present message, in line 9. If it is found, the offset value o_i of the present message is assigned such that it is transmitted at the starting point of the biggest empty interval, in line 10. In case of multiple empty intervals with the same length, the earliest one can be chosen. When the offset assignment is realized for the present message, it is removed from the Q_{mes} and added to the stack S_{proc}.

On the contrary, if there is not any available interval for the current message (line 13), the algorithm should try to create a space for it by changing the offsets of the messages

Algorithm 10 WTDMA Scheduling Algo - Heuristic

```
Input: $\mathcal{M}$ , $S = \{s_1, ..., s_m\}$
Output: $\mathcal{O} = \{o_1, ..., o_m\}$
Init: $o_i = \emptyset, i = 1, ..., m$; $H = gcd(p_1, ..., p_m)$ ; $L_s = lcm(p_1, ..., p_m)$ ; $W_{mes} = \emptyset$ ; $W_{dep,i} = \emptyset, i = 1, ..., m$ ; $Q_{mes} = \emptyset$ ; $S_{proc} = \emptyset$ .
Construct a queue $Q_{mes}$ of messages according to their periods
Construct the dependency lists $W_{dep,i}$ for all messages ($i = 1, ..., m$).
while $Q_{mes}$ is not empty do
    Pick the present message $M_i$ from $Q_{mes}$.
    Construct an imaginary timeline with the messages in $W_{dep,i}$.
    if Available room for the current message then
        Place the current message at the Starting Point $SP$ of the biggest empty interval. That is, $o_i = (s_i * L_s) + SP$.
        Remove the message from $Q_{mes}$.
        Add it to the processed message stack $S_{proc}$.
    else
        Find the starting point of the earliest empty interval.
        Find $M_{target}$ that comes right after the earliest empty interval.
        Construct a new queue $Q_{mes}^{new}$ such that the current message $M_i$ will come first.
        Right after $M_i$, all the previously processed messages until $M_{target}$ and $M_{target}$ will be transferred from the stack $S_{proc}$ to $Q_{mes}^{new}$. The offset values $o_i$ of those messages which are transferred from $S_{proc}$ will be updated as $\emptyset$ since they will required to be processed again.
        The remaining messages in $Q_{mes}$ will be transferred to $Q_{mes}^{new}$. Then replace $Q_{mes}$ with $Q_{mes}^{new}$.
    end if
end while
```

which are already processed. Firstly, the starting point of the earliest empty interval is found even if its length is smaller than the biggest empty interval (line 14). Then, the message M_{target} that comes right after the earliest empty interval in the imaginary timeline is found (line 15). Changing or cancelling the offset value of M_{target} may help to create an available interval for the present message since the length of the earliest empty interval will increase when M_{target} is removed. Therefore, between the lines 16 and 18, the offset assignments of all CAN messages except the messages that are processed before M_{target} is canceled. The offset assignments for the messages that are processed earlier than M_{target} are preserved and the other messages will be processed again such that the present message is handled first. Accordingly, the stack of the processed messages S_{proc} and the queue for the waiting messages Q_{mes} are updated.

Hereby, it has to be noted that the algorithm tries to find an offset assignment solution in an iterative way. When it does not find an empty space for the present message, it cancels some of the assigned offsets and goes back. That's why, a timeout should be applied while running the algorithm since it runs forever in case of a problem that has no solution.

8.4 Evaluation of the WTDMA Scheduling Methods

The proposed WTDMA scheduling methods are evaluated in this section with CAN message sets that are constructed in line with the real automotive applications. The message sets that consist of m CAN messages are randomly generated while evaluation. The message lengths are chosen randomly such that the payload of the message can change from 0 byte to 8 bytes. In order to compute the message length in terms of second, the bit rate is accepted as 250 kbps that is very common bit rate in practice. Moreover, the message periods are chosen randomly from the options: 5, 10, 20, 40, 80, 100, 200 and 1000 ms.

The evaluation is conducted in MATLAB R2018b [123] together with the CPLEX ILP solver 12.6 [120] which is used as embedded in the MATLAB environment. The host machine consists of Intel Core i3-2370M CPU @2.40 GHz with 8,00 GB RAM.

Table 8.4: Comparison of the success rate of the first step methods.

		H-S1	ILP-S1
m=30	SR(%)	99	99,5
m=40	SR(%)	88	100
m=50	SR(%)	72,5	98

In Table 8.4, the first step heuristic (H-S1) is evaluated in terms of its success rates (SR) by changing the number of messages in the message set. The success rate of ILP-S1 is given as a reference since ILP-S1 is able to find a solution if a solution exists. That is, the SR of ILP-S1 gives the ratio of the generated CAN message sets that have a solution to the total number of message sets. Furthermore, the evaluation is realized with 200 different message sets for each case and the bus load of a message set is forced to be higher than 50% to make the case more challenging. The success rate of H-S1 decreases from 99% to 72.5% when the number of messages m in a CAN message set is increased from 30 to 50. However, the ILP-S1 maintains its performance when m increases to 50 which shows that 98% of the generated message sets still have a solution when $m = 50$.

Furthermore, it has to be noted that the change in m directly affects the bus load during the evaluation. Thus, it is important to state that the bus load changes between 50% and 99,59% with an average of 64,24% when $m = 30$; between 50% and 98,86% with an average of 75,46% when $m = 40$; between 53,19% and 99,98% with an average of 85,04% when $m = 50$.

We next continue with the evaluation of H-S1 and ILP-S1 in terms of their runtimes. In Table 8.5, the runtimes of the first step methods are compared for the same set of CAN messages for which both methods can find solution. That is, we excluded the message sets which are solved only by ILP-S1. Although the runtimes are below 1 s , it is seen that H-S1 finds a solution slightly faster. Regarding the message sets which are solved only by ILP-S1, the maximum runtime is recorded as 0,063 s, 400,44 s and 1677 s (almost 30 min.) when m is 30, 40 and 50, respectively.

Additionally, it is observed that all of the CAN message sets that could not be solved

Table 8.5: Comparison of the run-times of the first step methods.

		H-S1	ILP-S1
m=30	mean(s)	0,0089	0,0392
	max(s)	0,0671	0,0630
m=40	mean(s)	0,0171	0,0554
	max(s)	0,0580	0,1560
m=50	mean(s)	0,0419	0,0679
	max(s)	0,0890	0,2810

by H-S1 have bus load higher than 90%. In order to verify that H-S1 provides a solution with higher success rate when the CAN message sets have bus load lower than 90%, we conducted an evaluation where m changes as 50, 60 and 70 and the bus load is forced to be between 50% and 90%. In this evaluation, it is observed that H-S1 finds solutions for all cases with SR of 100%. Moreover, the runtimes of H-S1 and ILP-S1 are measured very close to each other also in this evaluation and they are below 1 s.

To sum up, both first step methods provide similar performance when the bus load is lower than 90%. Both of them solves the first step problem with a success rate of 100% and in shorter than 1 s. On the contrary, when the bus load is more than 90%, the heuristic H-S1 rarely finds a solution although it takes shorter than 1 s. Moreover, ILP-S1 can take up to 0.5 hours for the case where $m = 50$, but it finds a solution if it exists for the problem. It has to be noted that the runtimes in the order of hours are acceptable for the WTDMA scheduling design since it is not repeated very frequently.

The solution of the first step is the input of the second step problem. Thus, the second step methods are evaluated naturally for the message sets which have feasible solutions after the first step. Therefore, we evaluate the success rate of the second step heuristic (H-S2) for the CAN message sets that have solutions after the first step when ILP-S1 is applied. After evaluation with several challenging CAN message sets, it is observed that both ILP-S2 and H-S2 always find a solution that means the success rate is 100% independently from the number of messages and the bus load. It has to be noted that, H-S2 is able to find a solution even for the CAN message sets whose bus loads are close to 100%.

Regarding the runtimes of ILP-S2 and H-S2, the comparison is presented in Table 8.6. It is seen from the table that ILP-S2 provides a solution less than 4 s when $m = 50$. Furthermore, it takes for H-S2 to find a solution less than 2.5 s when $m = 50$.

Table 8.6: The run-time comparison for H-S2 and ILP-S2.

		H-S2	ILP-S2
m=30	mean(s)	0,1905	0,0122
	max(s)	0,4597	0,0320
m=40	mean(s)	0,3837	0,0191
	max(s)	0,7744	0,3750
m=50	mean(s)	0,6644	0,0679
	max(s)	2,3793	3,9630

8.5 Discussion

In this chapter, WTDMA schedule design problem is discussed. The WTDMA model is introduced as the traffic shaping layer method of CANDS in this thesis. That's why, the schedule design methods compatible to WTDMA model are required within CANDS framework. We divide the scheduling problem into two steps such that the first step places the messages into mini time slots and the second step assigns ultimate offset values by respecting the mini time slot assignments which are decided at the end of the first step. Accordingly, ILP-based methods and heuristic methods are proposed for both steps and they are evaluated with several CAN message sets that are generated randomly in line with the practical applications.

With the comprehensive evaluation, it is verified that WTDMA scheduling of the practical CAN message sets can be realized in two steps with the proposed methods. Specifically, H-S1 finds solutions with high probability when the bus load is lower than 90%, but it is not very successful in finding a solution when the bus load of the message set is above 90%. Moreover, the process of H-S1 takes less than 1 s for all trials independently from its success in finding a solution. On the contrary, ILP-S1 is able to solve the first step mini time slot assignment problem for the cases where H-S1 is unsuccessful at the expense of higher runtimes. The runtimes of ILP-S1 and ILP-H1 look similar and less than 1 s for the problems where H-S1 is able to find a solution. However, the process of the ILP-S1 can be as high as 30 minutes to find a feasible solution for challenging CAN message sets. Thus, it makes sense to try to solve the first step problem with H-S1 before applying the ILP-S1. Regarding the second step, it is seen that a successful heuristic H-S2 is developed which is able to find solutions for all cases during the evaluation. Moreover, the runtimes of the second step methods ILP-S2 and H-S2 are similar and in the order of seconds. Thus, it is concluded that ILP-S2 and H-S2 provide comparable performance while solving the second step problem.

CHAPTER 9

CONCLUSION

Controller Area Network (CAN) is still the most widespread in-vehicle communication bus in modern cars today due to advantages such as reliability, low cost and simplicity. Furthermore, its importance is supported with the introduction of CAN with Flexible Data-rate (CAN-FD) in 2012 and the recent development of CAN Extended Length (CAN-XL). In particular, CAN is essential for safety critical automotive applications which require high real time Quality of Service (QoS) such as engine control, suspension control, traction control, and others. A possible shortcoming of CAN is its lack of a global time, which is required by safety-critical automotive applications that are realized on distributed nodes in order to coordinate their tasks. Hence, it is highly relevant to support clock synchronization (CS) on CAN. Additionally, the realization of CS on CAN makes it possible to provide deterministic medium access on CAN, that is highly desired in order to increase the bandwidth efficiency by reducing the message latencies. In brief, having CS on CAN is notable both for the synchronization of the distributed nodes and for CAN protocol itself with its contribution to the deterministic bus access.

This thesis proposes a general framework CANDS (CAN with Determinism and Synchronization support) for in-vehicle network protocols. CANDS features a hierarchy of novel clock synchronization (CS) algorithms with different levels of clock accuracy and implementation complexity. Moreover, CANDS realizes our original idea of weak time division multiple access (WTDMA) for deterministic bus access. In particular, WTDMA can be implemented in software and hence it is fully compatible with the standard CAN protocol and its extensions. Furthermore, WTDMA does not depend on a specific CS method but it can operate with different levels of clock accuracy that are achieved by the developed CS methods at the expense of different levels of complexity within CANDS. In order to efficiently use WTDMA, the thesis further develops scheduling methods for assigning time slots in WTDMA.

Considering CS methods for CAN, periodic timestamps which are expected to be taken at the same time by all nodes are generally used. In addition, a TM whose local clock is assumed as the perfect clock broadcasts its timestamp by sending a RM to slave nodes. The slave nodes correct their local clocks periodically whenever they receive timestamps of TM in RMs. In this thesis, Improved Software-based clock synchronization (ISCS) method is proposed which brings advantages over the leading CS schemes, AUTOSAR CS and Gergeleit's method in the literature. AUTOSAR CS method provides the best clock accuracy performance at the expense of higher bandwidth usage for CS. However, Gergeleit's method requires the half of the bandwidth of AUTOSAR CS method while it can not achieve the clock accuracy performance of AUTOSAR CS method. As a new idea, our ISCS method benefits from the ongoing CAN message transmission on the bus. Thus, ISCS achieves the clock accuracy performance of the AUTOSAR CS method, while requiring the bandwidth for CS as low as Gergeleit's method. In brief, ISCS scheme is able to provide the strong features of two leading CS scheme at the same time with an original idea that depends on the usage of CAN message traffic between the nodes. On the other hand, CS schemes define the simultaneous timestamping instants and transmission of timestamps within periodic RMs and hence offset correction mechanism. In order to have better clock accuracy results, slave nodes should perform drift correction between RMs in addition to offset correction when receiving a RM. Accordingly, several original methods which enable drift correction on CAN in line with the existing CS schemes are proposed in this thesis. Firstly, the basic drift estimation approach by using the periodic timestamps within RMs is formulated. Secondly, the periodic timestamps are evaluated with a discrete-time feedback control loop in order to estimate the clock drift. Thirdly, a novel drift estimation approach that does not depend on the timestamps is introduced by using the internal re-synchronization mechanism of the CAN bit timing. That is, our bit timing based drift estimation method is not affected by the TS inaccuracies and hence provide better drift estimation performance. However, the proposed drift estimation methods are developed to supplement the offset correction applications in order to avoid the clock drift between RMs. Therefore, the resultant CS methods for CAN depend on TS quality which is determined by the difference between the time instants when timestamps are taken on different nodes. Although it is well known for CS applications that the clock accuracy performance depends on TS quality, there is not any study in the literature which evaluates and compares the existing TS methods on CAN. Thus, this thesis further explores the existing TS methods in detail and proposes a new TS method which mitigates the identified uncertainties of the existing TS methods. That is, our new predictable TS method enables timestamps with higher quality on CAN. As a consequence, the clock accuracies below 100 ns is achieved with comprehensive hardware experiments when our novel timestamping method and CS algorithms perform together.

The CS methods in this thesis offer different levels of clock accuracy with different levels of implementation complexity. Benefiting from these CS methods, this thesis further introduces Weak Time Division Multiple Access (WTDMA) as a new slotted medium access on CAN that is compatible with the different levels of clock accuracy. Moreover, the proposed WTDMA takes also inevitable software delays into account and hence can be realized on software without requiring any modification of the existing CAN controllers. Particularly, WTDMA model is presented with the sufficient conditions that define the minimum window size according to the achievable clock accuracy and the maximum software delay for the correct operation of WTDMA. Therefore, WTDMA can be extended according to the specific network parameters for example the bit rate and the message payloads which determine the message length together. In addition, WTDMA does not use guard times differently from the conventional Time Division Multiple Access (TDMA) approaches for example in wireless networks. Due to the specific properties of CAN protocol such as Carrier Sense Multiple Access/Collision Resolution (CSMA/CR) and the non-preemptive message transmission, any interference from the previous or the next CAN message is tolerated in WTDMA. That is, the guard times are not required in WTDMA which makes higher bandwidth utilization possible. As a consequence, the message response times (RTs) are deterministic and mostly determined by the message transmission times in WTDMA. In this thesis, it is experimentally shown that bus loads above 90% are achieved in WTDMA while the message latencies are in the order of hundreds of microseconds and almost equal to the message transmission times.

Having introduced the WTDMA model, this thesis also develops new message scheduling algorithms for WTDMA, which assign the given CAN messages into the time slots. The proposed Integer Linear Programming (ILP) methods and heuristic WTDMA scheduling methods are evaluated with several CAN message sets that are constructed in line with the practical automotive applications. It is shown that the developed scheduling methods for WTDMA are able to find feasible message assignments that can be followed in a successful WTDMA operation.

Last but not least, it has to be noted that the advancements in this thesis have potential to be used by timing-based intrusion detection systems for the security of CAN, which is left as a future work. Specifically, our bit timing based drift estimation approach can be used to identify the CAN nodes within the system since each CAN node has an individual clock drift. Accordingly, the malicious attacks that are realized by an attacker which transmits intrusion CAN messages on the bus can be determined since the clock drift of the attacker can not be the same with the nodes in the original system. Furthermore, our predictable TS method likely improves the performances of timing-based intrusion detection systems by providing more precise timestamps. In addition, future work will be concerned with the extension of our CS methods to fault-tolerant CS.

REFERENCES

[1] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin, "Intra-vehicle networks: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 534–545, 2015.

[2] R. Wang, J.-J. Ma, C.-S. Chen, B.-Z. Wang, and J. Xiong, "Low-profile implementation of u-shaped power quasi-isotropic antennas for intra-vehicle wireless communications," *IEEE Access*, vol. 8, pp. 48557–48565, 2020.

[3] E. Choi, H. Song, S. Kang, and J.-W. Choi, "High-speed, low-latency in-vehicle network based on the bus topology for autonomous vehicles: Automotive networking and applications," *IEEE Vehicular Technology Magazine*, vol. 17, no. 1, pp. 74–84, 2022.

[4] ISO, "ISO-11898, Road Vehicles-Interchange of Digital Information – Controller Area Network (CAN) for High-Speed Communication," 1993.

[5] "FlexRay communication system, protocol specification, version 2.0.," June 2004.

[6] "LIN specification package, revision 2.0." LIN Consortium, 2003.

[7] "MOST specification revision 2.3." MOST Cooperation, 2008.

[8] C. Gao, G. Wang, W. Shi, Z. Wang, and Y. Chen, "Autonomous driving security: State of the art and challenges," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7572–7595, 2022.

[9] F. A. Butt, J. N. Chattha, J. Ahmad, M. U. Zia, M. Rizwan, and I. H. Naqvi, "On the integration of enabling wireless technologies and sensor fusion for next-generation connected and autonomous vehicles," *IEEE Access*, vol. 10, pp. 14643–14668, 2022.

[10] A. Valenzano and G. Cena, "Controller area networks for embedded systems," in *Networked Embedded Systems* (R. Zurawsk, ed.), pp. 15–1–15–38, CRC Press, 2009.

[11] J. Huang, M. Zhao, Y. Zhou, and C.-C. Xing, "In-vehicle networking: Protocols, challenges, and solutions," *IEEE Network*, vol. 33, no. 1, pp. 92–98, 2019.

[12] J. Wang, J. Liu, and N. Kato, "Networking and communications in autonomous driving: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1243–1274, 2019.

[13] T. ElBatt, C. Saraydar, M. Ames, and T. Talty, "Potential for intra-vehicle wireless automotive sensor networks," in *2006 IEEE Sarnoff Symposium*, pp. 1–4, 2006.

[14] A. Neumann, M. J. Mytych, D. Wesemann, L. Wisniewski, and J. Jasperneite, "Approaches for in-vehicle communication – an analysis and outlook," in *Computer Networks* (P. Gaj, A. Kwiecień, and M. Sawicki, eds.), (Cham), pp. 395–411, Springer International Publishing, 2017.

[15] W. Steiner, "TTEthernet specification." TTTech Computertechnik AG, Nov. 2008.

[16] F. Hartwich, "CAN with flexible data rate," *The international CAN Conference (iCC)*, pp. 1–9, 2012.

[17] M. Akpinar, K. W. Schmidt, and E. G. Schmidt, "Improved clock synchronization algorithms for the controller area network (CAN)," in *International Conference on Computer Communication and Networks*, pp. 1–8, IEEE, 2019.

[18] M. Akpınar, E. G. Schmidt, and K. Werner Schmidt, "Drift correction for the software-based clock synchronization on controller area network," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, 2020.

[19] S. Einspieler, N. Rathakrishnan, A. Prabhakara, B. Steinwender, and W. Elmenreich, "High accuracy software-based clock synchronization over can," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–9, 2021.

[20] F. Luckinger and T. Sauter, “Software-based autosar-compliant precision clock synchronization over can,” *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2022.

[21] L. B. Othmane, L. Dhulipala, M. Abdelkhalek, N. Multari, and M. Govindarasu, “On the performance of detecting injection of fabricated messages into the can bus,” *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 468–481, 2022.

[22] K. Agrawal, T. Alladi, A. Agrawal, V. Chamola, and A. Benslimane, “Novelads: A novel anomaly detection system for intra-vehicular networks,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2022.

[23] Y. Zhao, Y. Xun, and J. Liu, “ClockIDS: A real-time vehicle intrusion detection system based on clock skew,” *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[24] A. Michaels, V. S. S. Palukuru, M. J. Fletcher, C. Henshaw, S. Williams, T. Krauss, J. Lawlis, and J. Moore, “CAN bus message authentication via co-channel rf watermark,” *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2022.

[25] U. Ezeobi, H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, “Reverse engineering controller area network messages using unsupervised machine learning,” *IEEE Consumer Electronics Magazine*, vol. 11, no. 1, pp. 50–56, 2022.

[26] T. Führer, B. Müller, W. Dieterle, F. Hartwich, R. Hugel, and M. Walther, “Time triggered communication on CAN,” in *International CAN Conference*, pp. 1–6, 2000.

[27] “AUTOSAR Release 4.3.1 – Specification of Time Synchronization over CAN.” in AUTOSAR Std (Release 4.1.1), July 2015.

[28] M. Gergeleit and H. Streich, “Implementing a distributed high-resolution real-time clock using the CAN-bus,” in *International CAN-Conference*, 1994.

[29] M. Akpınar, E. G. Schmidt, and K. W. Schmidt, “Evaluation of clock synchronization algorithms for controller area network,” in *Signal Processing and Communications Applications Conference*, pp. 1–4, May 2020.

[30] D. E. Arkadaş, M. Akpınar, E. G. Schmidt, and K. W. Schmidt, "Clock synchronization for the controller area network using bit timing information," in *Signal Processing and Communications Applications Conference*, pp. 1–4, May 2022.

[31] M. Akpinar, E. G. Schmidt, and K. W. Schmidt, "Highly accurate clock synchronization with drift correction for the controller area network," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–1, 2022.

[32] Y. He, Z. Jia, M. Hu, C. Cui, Y. Cheng, and Y. Yang, "The hybrid similar neighborhood robust factorization machine model for CAN bus intrusion detection in the in-vehicle network," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–9, 2021.

[33] F. Amato, L. Coppolino, F. Mercaldo, F. Moscato, R. Nardone, and A. Santone, "Can-bus attack detection with deep learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5081–5090, 2021.

[34] R. B. GmbH, "CAN XL – the next step in CAN evolution." `https://www.bosch-semiconductors.com/media/ip_modules/pdf_2/can_xl_1/canxl_intro_20210225.pdf`, 2021. Accessed on 13.06.2022.

[35] H. Daigmorte, M. Boyer, and J. Migge, "Reducing CAN latencies by use of weak synchronization between stations," in *Proceedings of the 16th International CAN Conference*, pp. 1–8, 2017.

[36] M. Hadded, P. Muhlethaler, A. Laouiti, R. Zagrouba, and L. A. Saidane, "TDMA-Based MAC protocols for vehicular ad hoc networks: A survey, qualitative analysis, and open research issues," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2461–2492, 2015.

[37] O. Jubran and B. Westphal, "Optimizing guard time for tdma in a wireless sensor network - case study," in *39th Annual IEEE Conference on Local Computer Networks Workshops*, pp. 597–601, 2014.

[38] M. Taralkar, "Computation of CAN bit timing parameters simplified," in *CAN in Automation, iCC*, 2012.

[39] "CiA 601-4 version 2.1.0 - CAN FD node and system design – part 4: Signal improvement," 2021.

[40] F. Luckinger and T. Sauter, "Software-based AUTOSAR-compliant precision clock synchronization over CAN," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2022.

[41] H. Kim, X. Ma, and B. R. Hamilton, "Tracking low-precision clocks with time-varying drifts using Kalman filtering," *IEEE/ACM Transactions on Networking*, vol. 20, pp. 257–270, Feb 2012.

[42] K. S. Yildirim and A. Kantarci, "External gradient time synchronization in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 633–641, 2014.

[43] G. Rodriguez-Navas, S. Roca, and J. Proenza, "Orthogonal, fault-tolerant, and high-precision clock synchronization for the controller area network," *IEEE Transactions on Industrial Informatics*, vol. 4, pp. 92–101, May 2008.

[44] S. Park, H. Kim, H. Kim, C. N. Cho, and J. Choi, "Synchronization improvement of distributed clocks in EtherCAT networks," *IEEE Communications Letters*, vol. 21, pp. 1277–1280, June 2017.

[45] "Cardinal Components Inc. applications brief no. A.N. 419." `http://www.cardinalxtal.com/static/frontend/files/cardinal-measuring-clock-stability.pdf`, 1998. Accessed on 13-06-2022.

[46] Z. Yang, L. He, L. Cai, and J. Pan, "Temperature-assisted clock synchronization and self-calibration for sensor networks," *IEEE Transactions on Wireless Communications*, vol. 13, pp. 3419–3429, June 2014.

[47] B. Martinez, X. Vilajosana, and D. Dujovne, "Accurate clock discipline for long-term synchronization intervals," *IEEE Sensors Journal*, vol. 17, pp. 2249–2258, April 2017.

[48] R. Baldoni, A. Corsaro, L. Querzoni, S. Scipioni, and S. Tucci Piergiovanni, "Coupling-based internal clock synchronization for large-scale dynamic dis-

tributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 5, pp. 607–619, 2010.

[49] J. Liu, Z. Zhou, Z. Peng, J. Cui, M. Zuba, and L. Fiondella, "Mobi-sync: Efficient time synchronization for mobile underwater sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 2, pp. 406–416, 2013.

[50] Z. Gu, G. Han, H. Zeng, and Q. Zhao, "Security-aware mapping and scheduling with hardware co-processors for FlexRay-based distributed embedded systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 3044–3057, 2016.

[51] K. Schmidt and E. G. Schmidt, "Systematic message schedule construction for time-triggered CAN," *Vehicular Technology, IEEE Transactions on*, vol. 56, no. 6, pp. 3431–3441, 2007.

[52] A. Batur, E. G. Schmidt, and K. W. Schmidt, "Evaluation of response time distributions for controller area network messages," in *Signal Processing and Communications Applications Conference*, pp. 1–4, 2018.

[53] "IEEE standard for a precision clock synchronization protocol for networked measurement and control systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–269, 2008.

[54] G. Breaban, M. Koedam, S. Stuijk, and K. Goossens, "Time synchronization for an emulated CAN device on a multi-processor system on chip," *Microprocess. Microsyst.*, vol. 52, pp. 523–533, July 2017.

[55] G. Cena, I. C. Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "Evaluation of EtherCAT distributed clock performance," *IEEE Transactions on Industrial Informatics*, vol. 8, pp. 20–29, Feb 2012.

[56] "IEEE standard for local and metropolitan area networks - timing and synchronization for time - sensitive applications in bridged local area networks," *IEEE Std 802.1AS-2011*, pp. 1–292, 2011.

[57] H. Lim, D. Herrscher, L. Völker, and M. J. Waltl, "IEEE 802.1AS time synchronization in a switched Ethernet based in-car network," in *IEEE Vehicular Networking Conference*, pp. 147–154, 2011.

[58] "CiA 603 version 1.0.0 - CAN network time management," 2017.

[59] "SAM D5x/E5x family data sheet." `https://www.microchip.com`. Accessed on 13-06-2022.

[60] "STM32F429/439 advanced Arm ® -based 32-bit MCUs reference manual." `https://www.st.com/resource/en/reference_manual/rm0090-stm32f405415-stm32f407417-stm32f427437-and\-stm32f429439-advanced-armbased-32bit-mcus-\stmicroelectronics.pdf`. Accessed on 13-06-2022.

[61] "XMC4700/XMC4800 microcontroller series for industrial applications,." `https://www.infineon.com/dgdl/Infineon-XMC4700-XMC4800-DS-v01_01-EN.pdf?fileId=5546d462518ffd850151908ea8db00b3`. Accessed on 16-06-2022.

[62] K. Turski, "A global time system for CAN," in *International CAN Conference*, pp. 1–6, 1994.

[63] B. Donnelly and J. Cosgrove, "Achieving microsecond accuracy with 32 bit microcontrollers using the controller area network (CAN)," in *IET Conference Proceedings*, pp. 508–513, January 2004.

[64] D. Lee and J. Allan, "Fault-tolerant clock synchronisation with microsecond-precision for CAN networked systems," in *International CAN Conference*, pp. 1–6, 2003.

[65] "Fujitsu FlexRay Evaluation Board: SK-91465X-100PMC." `https://www.fujitsu.com/downloads/MICRO/fma/mcu/ug-910056-11-sk-91465x-100pmc.pdf`, 2008. Accessed on 13-06-2022.

[66] "Zedboard." `http://zedboard.org/product/zedboard`. Accessed on 13-06-2022.

[67] “Zynqberry Evaluation Board.” `https://www.digikey.com/en/product-highlight/t/trenz/te0726-zynq-zynqberry-z-7010-module`, 2016. Accessed on 13-06-2022.

[68] C. Pang, J. Yan, and V. Vyatkin, “Time-complemented event-driven architecture for distributed automation systems,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 8, pp. 1165–1177, 2015.

[69] H. Sun, J. Sun, and J. Chen, “Quantized control of networked control systems under stochastic clock offsets,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 5, pp. 3004–3013, 2021.

[70] H. Mun, K. Han, and D. Lee, “Ensuring safety and security in can-based automotive embedded systems: A combination of design optimization and secure communication,” *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 7078–7091, 2020.

[71] M. L. Han, B. I. Kwak, and H. K. Kim, “Event-triggered interval-based anomaly detection and attack identification methods for an in-vehicle network,” *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2941–2956, 2021.

[72] H. J. Jo and W. Choi, “A survey of attacks on controller area networks and corresponding countermeasures,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–19, 2021.

[73] G. Xie, L. T. Yang, Y. Yang, H. Luo, R. Li, and M. Alazab, “Threat analysis for automotive can networks: A gan model-based intrusion detection technique,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4467–4477, 2021.

[74] K.-T. Cho and K. G. Shin, “Fingerprinting electronic control units for vehicle intrusion detection,” in *25th USENIX Security Symposium (USENIX Security 16)*, (Austin, TX), pp. 911–927, USENIX Association, Aug. 2016.

[75] J. Zhou, G. Xie, S. Yu, and R. Li, “Clock-based sender identification and attack

detection for automotive can network," *IEEE Access*, vol. 9, pp. 2665–2679, 2021.

[76] X. Ying, S. U. Sagong, A. Clark, L. Bushnell, and R. Poovendran, "Shape of the cloak: Formal analysis of clock skew-based intrusion detection system in controller area networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2300–2314, 2019.

[77] H. Ji, Y. Wang, H. Qin, X. Wu, and G. Yu, "Investigating the effects of attack detection for in-vehicle networks based on clock drift of ecus," *IEEE Access*, vol. 6, pp. 49375–49384, 2018.

[78] P.-S. Murvay and B. Groza, "TIDAL-CAN: Differential timing based intrusion detection and localization for controller area network," *IEEE Access*, vol. 8, pp. 68895–68912, 2020.

[79] F. Hartwich, "Can frame time-stamping supporting autosar time base synchronization," *The international CAN Conference (iCC)*, pp. 1–5, 2017.

[80] R. B. GmbH, "M-CAN add-ons." White Paper, 15.03.2019. Accessed on 06-02-2022.

[81] R. Davis, A. Burns, R. Bril, and J. Lukkien, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Syst.*, vol. 35, pp. 239–272, April 2007.

[82] H. Daigmorte and M. Boyer, "Traversal time for weakly synchronized CAN bus," in *International Conference on Real-Time Networks and Systems*, pp. 35–44, 2016.

[83] S. Mubeen, J. Maki-Turja, and M. Sjodin, "Worst-case response-time analysis for mixed messages with offsets in controller area network," in *Emerging Technologies Factory Automation, IEEE Conference on*, pp. 1–10, 2012.

[84] Y. Chen, R. Kurachi, H. Takada, , and G. Zeng, "Schedulability comparison for CAN message with offset: Priority queue versus FIFO queue," *RTNS*, pp. 181–192, 2011.

[85] H. Daigmorte and M. Boyer, “Evaluation of admissible can bus load with weak synchronization mechanism,” in *Proceedings of the 25th International Conference on Real-Time Networks and Systems - RTNS ’17*, (Grenoble, FR), pp. 277–286, 2017.

[86] K. Tindell and A. Burns, “Guaranteeing message latencies on controller area network (CAN),” in *In Proceedings of the 1st International CAN Conference*, pp. 1–2, CiA, 1994.

[87] S. Mubeen, J. Mäki-Turja, and M. Sjödin, “Extending worst case response-time analysis for mixed messages in controller area network with priority and FIFO queues,” *IEEE Access*, vol. 2, pp. 365–380, 2014.

[88] K. Tindell, A. Burns, and A. Wellings, “Calculating controller area network (CAN) message response times,” *Control Engineering Practice*, vol. 3, pp. 1163–1169, 1995.

[89] N. Audsley, “On priority assignment in fixed priority scheduling,” *Information Processing Letters*, vol. 79, no. 1, pp. 39–44, 2001.

[90] R. Davis, S. Kollmann, V. Pollex, and F. Slomka, “Controller area network (CAN) schedulability analysis with FIFO queues,” in *Real-Time Systems (ECRTS), 2011 23rd Euromicro Conference on*, pp. 45–56, July 2011.

[91] K. W. Schmidt, “Robust priority assignments for extending existing controller area network applications,” *IEEE Transactions on Industrial Informatics*, vol. 10, pp. 578–585, Feb 2014.

[92] M. Grenier, L. Havet, and N. Navet, “Pushing the limits of CAN - scheduling frames with offsets provides a major performance boost,” in *European Congress on Embedded Real Time Software*, 2008.

[93] L. Du and G. Xu, “Worst case response time analysis for CAN messages with offsets,” in *Vehicular Electronics and Safety, IEEE International Conference on*, pp. 41–45, 2009.

[94] P. Yomsi, D. Bertrand, N. Navet, and R. Davis, “Controller area network (CAN): Response time analysis with offsets,” in *Factory Communication Systems, IEEE International Workshop on*, pp. 43–52, 2012.

[95] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Worst-case response-time analysis for mixed messages with offsets in controller area network," in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, pp. 1–10, 2012.

[96] S. Jiang, *Medium Access Control (MAC)*, pp. 51–74. Singapore: Springer Singapore, 2018.

[97] C. Easttom, *An In-Depth Guide to Mobile Device Forensics*. CRC Press, 2021.

[98] L. S. Cerqueira, A. B. Vieira, L. F. Vieira, M. A. Vieira, and J. A. Nacif, "A cooperative protocol for pervasive underwater acoustic networks," *Wireless Networks*, vol. 27, no. 3, pp. 1941–1963, 2021.

[99] C. Zhai, Z. Zou, Q. Chen, L. Zheng, and H. Tenhunen, "Optimization on guard time and synchronization cycle for TDMA-based deterministic RFID system," in *2015 IEEE International Conference on RFID Technology and Applications (RFID-TA)*, pp. 71–75, 2015.

[100] G. Leen and D. Heffernan, "Ttcan: a new time-triggered controller area network," *Microprocessors and Microsystems*, vol. 26, no. 2, pp. 77 – 94, 2002.

[101] N. Navet and F. Simonot-Lion, "In-vehicle communication networks: A historical perspective and review," in *Industrial Communication Technology Handbook*, pp. 50–1, CRC Press, 2017.

[102] W. Lawrenz, *Can System Engineering: From Theory to Practical Applications*. Springer New York, 2013.

[103] G. Cena, I. Cibrario Bertolotti, T. Hu, and A. Valenzano, "On a software-defined can controller for embedded systems," *Computer Standards & Interfaces*, vol. 63, pp. 43–51, 2019.

[104] L. Rodrigues, M. Guimaraes, and J. Rufino, "Fault-tolerant clock synchronization in CAN," in *IEEE Real-Time Systems Symposium*, pp. 420–429, Dec 1998.

[105] A. Mahmood, R. Exel, and T. Sauter, "Impact of hard-and software timestamping on clock synchronization performance over IEEE 802.11," in *2014 10th*

IEEE Workshop on Factory Communication Systems (WFCS 2014), pp. 1–8, 2014.

[106] A. Mahmood, R. Exel, H. Trsek, and T. Sauter, "Clock synchronization over ieee 802.11—a survey of methodologies and protocols," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 907–922, 2017.

[107] B. J. Choi, H. Liang, X. Shen, and W. Zhuang, "DCS: Distributed asynchronous clock synchronization in delay tolerant networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 3, pp. 491–504, 2012.

[108] K. S. Yildirim and A. Kantarci, "Time synchronization based on slow-flooding in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 244–253, 2014.

[109] F. Tirado-Andrés and A. Araujo, "Performance of clock sources and their influence on time synchronization in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 15, no. 9, pp. 1–16, 2019.

[110] N. Howgrave-Graham, "Approximate integer common divisors," in *International Cryptography and Lattices Conference*, pp. 51–66, Springer, 2001.

[111] J. Yu and P.-Z. Lu, "AGCD: a robust periodicity analysis method based on approximate greatest common divisor," *Frontiers of Information Technology & Electronic Engineering*, vol. 16, no. 6, pp. 466–473, 2015.

[112] J. Yu, P. Lu, J. Han, and J. Lu, "Detecting regularities of traffic signal timing using GPS trajectories," *IEICE Transactions on Information and Systems*, vol. 101, no. 4, pp. 956–963, 2018.

[113] Y. Chen and P. Q. Nguyen, "Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 502–519, Springer, 2012.

[114] P. Hank and E. Jöhnk, *SJA1000 Stand-alone CAN controller.* Philips Semiconductors, Dec. 1997.

[115] "TTCAN module." `https://www.infineon.com/dgdl/ap2900612_TTCAN_GettingStarted.pdf?fileId=db3a30431c69a49d011cd7006a4d1316`. Accessed on 13-06-2022.

[116] N. Navet, Y.-Q. Song, and F. Simonot, "Worst-case deadline failure probability in real-time applications distributed over controller area network," *Journal of Systems Architecture*, vol. 46, pp. 607 – 617, 2000.

[117] J. Fonseca, F. Coutinho, and J. Barreiros, "Scheduling for a TTCAN network with a stochastic optimization algorithm," in *International CAN in Automation Conference*, 2002.

[118] T. Nolte, H. Hansson, C. Norstrom, and S. Punnekkat, "Using bit-stuffing distributions in can analysis," *IEEE Real-Time Embedded Systems Workshop*, Dec 2001.

[119] M. Short, I. Sheikh, S. Aley, and I. Rizvi, "Bandwidth-efficient burst error tolerance in TDMA-based CAN networks," in *ETFA2011*, pp. 1–8, 2011.

[120] IBM, "ILOG CPLEX optimization studio v12.8."

[121] M. Çağlar Güldiken, E. Güran Schmidt, and K. Werner Schmidt, "Telegram scheduling for the multifunction vehicle bus (MVB): Algorithms and evaluation," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, 2020.

[122] F. Sagstetter, M. Lukasiewycz, and S. Chakraborty, "Generalized asynchronous time-triggered scheduling for FlexRay," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 2, pp. 214–226, 2017.

[123] MATLAB, *version 9.5.0 (R2018a)*. Natick, Massachusetts: The MathWorks Inc., 2018.

[124] M. Taralkar, "Computation of CAN bit timing parameters simplified," in *Proceedings of the 11th International CAN Conference*, pp. 13–12–13–19, 2012.

Appendix A

PROOFS IN CHAPTER 5

A.1 Clock Accuracy of Offset Correction

We prove Theorem 1 from the main document, which is recalled here for convenience.

Theorem 1 *Consider the offset correction at time* t_k *according to* (5.1) *for a slave node Si with drift* d_{Si}*, software delays* J_{TM}*,* J_{Si}*, maximum TRI error* E_{TRI} *and the uncontrolled time duration* $t_{unc} = t_{\mathrm{RM}}$*. Then, the clock accuracy of node Si's LC is bounded by*

$$\max_t\{|\Delta c_{\mathrm{Si}}(t)|\} \leq C_{\mathrm{OC}} + \left\lceil \frac{|d_{\mathrm{Si}}| \cdot T}{T_{\mathrm{Si}}} \right\rceil \cdot T_{\mathrm{Si}}, \tag{A.1}$$

with the maximum clock difference after offset correction at the update times t_k^+

$$C_{\mathrm{OC}} = \max\{|\Delta N_{\mathrm{Si}}^{\min}|, |\Delta N_{\mathrm{Si}}^{\max}|\} \cdot T_{\mathrm{Si}} \tag{A.2}$$

and

$$\begin{aligned}
\Delta N_{\mathrm{Si}}^{\min} &= \left\lfloor \frac{t_{\mathrm{unc}} \cdot d_{\mathrm{Si}}}{T_{\mathrm{Si}}} \right\rfloor - \left\lceil \frac{E_{\mathrm{TRI}} + J_{\mathrm{Si}}}{T_{\mathrm{Si}}/(1 + d_{\mathrm{Si}})} \right\rceil - 1 \\
\Delta N_{\mathrm{Si}}^{\max} &= \left\lceil \frac{t_{\mathrm{unc}} \cdot d_{\mathrm{Si}}}{T_{\mathrm{Si}}} \right\rceil + \left\lceil \frac{J_{\mathrm{TM}}}{T_{\mathrm{Si}}} \right\rceil + \left\lceil \frac{E_{\mathrm{TRI}}}{T_{\mathrm{Si}}/(1 + d_{\mathrm{Si}})} \right\rceil,
\end{aligned}$$

Proof 1 *Assuming perfect timestamping at the reception time* $t_k - t_{\mathrm{unc}}$ *of the SYNC message and no software delay, we can evaluate the clock difference* $\Delta c_{\mathrm{Si}}((t_k - t_{\mathrm{unc}})^+)$ *when applying the offset correction in* (5.1) *at time* $t_k - t_{\mathrm{unc}}$ *(before the transmission of the RM). That is, using* (5.1) *and the perfect timestamps* $\tilde{t}_{\mathrm{TM},k}$ *and* $\tilde{N}_{\mathrm{Si},k}$ *taken by TM and Si, respectively, at* $t_k - t_{\mathrm{unc}}$*, we have*

$$N_{\mathrm{Si}}((t_k - t_{\mathrm{unc}})^+) = N_{\mathrm{Si}}((t_k - t_{\mathrm{unc}})^-) + \left\lfloor \frac{\tilde{t}_{\mathrm{TM},k}}{T_{\mathrm{Si}}} \right\rfloor - \tilde{N}_{\mathrm{Si},k}. \tag{A.3}$$

Considering the corresponding clock difference, we compute

$$\begin{aligned}\Delta c_{\mathrm{Si}}((t_k - t_{\mathrm{unc}})^+) =& \big(N_{\mathrm{Si}}((t_k - t_{\mathrm{unc}})^-) + \Big\lfloor \frac{\tilde{t}_{\mathrm{TM},k}}{T_{\mathrm{Si}}} \Big\rfloor \\ &- \tilde{N}_{\mathrm{Si},k}\big) \cdot T_{\mathrm{Si}} - \tilde{t}_{\mathrm{TM},k} \\ =& \Big\lfloor \frac{\tilde{t}_{\mathrm{TM},k}}{T_{\mathrm{Si}}} \Big\rfloor \cdot T_{\mathrm{Si}} - \tilde{t}_{\mathrm{TM},k}. \end{aligned} \tag{A.4}$$

Noting that $-y < \lfloor \frac{x}{y} \rfloor \cdot y - x \leq 0$ *for* $x \geq y > 0$ *and* $\tilde{t}_{\mathrm{TM},k} \geq T_{\mathrm{Si}}$*, we conclude that*

$$-T_{\mathrm{Si}} < \Delta c_{\mathrm{Si}}((t_k - t_{\mathrm{unc}})^+) \leq 0 \tag{A.5}$$

Next, we consider the effect of the TRI error, the software delay and the time difference between the timestamping instant $t_k - t_{\mathrm{unc}}$ *and the time of the offset correction* t_k.

Due to the TRI error, Si potentially takes the timestamp at a slightly different time than the TM. This TRI error is given by two components. First, there is a potential misalignment of the TQs of TM and Si (maximum one TQ duration). Second, any of the receiver nodes in the network can create the falling edge of the ACK bit also for the SYNC message. That is, depending on the distance of Si and TM from the node generating the falling edge, it is possible to observe an additional inaccuracy, whose upper bound is the propagation delay $d_{\mathrm{P}} = 5\,\mathrm{ns/m} \cdot L_{\mathrm{CAN}}$ *with the maximum length* L_{CAN} *between two CAN nodes [124]. In addition, due to variable software delays, TM and Si take the actual timestamp with a delay between* 0 *and* J_{TM} *and* 0 *and* J_{Si}*, respectively. Finally, the clock difference changes by* $t_{\mathrm{unc}} \cdot d_{\mathrm{Si}}$ *between* $t_k - t_{\mathrm{unc}}$ *and* t_k *due to the drift of Si's LC.*

Together, the actual timestamp $\hat{t}_{\mathrm{TM},k}$ *received from the TM varies between* $\tilde{t}_{\mathrm{TM},k}$ *(zero software delay) and* $\tilde{t}_{\mathrm{TM},k} + J_{\mathrm{TM}}$ *(maximum software delay). Similarly, the corresponding timestamp* $\hat{N}_{\mathrm{Si},k}$ *(in terms of HR ticks) taken by Si varies between* $\tilde{N}_{\mathrm{Si},k} - \Big\lceil \frac{E_{\mathrm{TRI}}}{T_{\mathrm{Si}}/(1+d_{\mathrm{Si}})} \Big\rceil$ *(zero software delay and negative TRI error) and* $\tilde{N}_{\mathrm{Si},k} + \Big\lceil \frac{E_{\mathrm{TRI}} + J_{\mathrm{Si}}}{T_{\mathrm{Si}}/(1+d_{\mathrm{Si}})} \Big\rceil$ *(maximum software delay and positive TRI error). Hereby,* $T_{\mathrm{Si}}/(1 + d_{\mathrm{Si}})$ *is the actual duration of Si's HR tick. Moreover, an additional clock difference of* $\frac{t_{\mathrm{unc}} \cdot d_{\mathrm{Si}}}{T_{\mathrm{Si}}}$ *is accumulated between* $t_k - t_{\mathrm{unc}}$ *and* t_k *due to the drift* d_{Si}.

Applying the offset correction

$$N_{\mathrm{Si}}((t_k)^+) = N_{\mathrm{Si}}((t_k)^-) + \Big\lfloor \frac{\hat{t}_{\mathrm{TM},k}}{T_{\mathrm{Si}}} \Big\rfloor - \hat{N}_{\mathrm{Si},k}. \tag{A.6}$$

with the actual timestamps $\hat{t}_{\mathrm{TM},k}$ *and* $\hat{N}_{\mathrm{Si},k}$*, the possible negative HR tick difference after offset correction at time* t_k *(smallest value of* $\hat{t}_{\mathrm{TM},k}$*, smallest value of* Δc_{Si} *term in* (A.5) *and largest value of* $\hat{N}_{\mathrm{Si},k}$*) is*

$$\Delta N_{\mathrm{Si}}^{\min} = \left\lfloor \frac{t_{\mathrm{unc}} \cdot d_{\mathrm{Si}}}{T_{\mathrm{Si}}} \right\rfloor - \left\lceil \frac{E_{\mathrm{TRI}} + J_{\mathrm{Si}}}{T_{\mathrm{Si}}/(1+d_{\mathrm{Si}})} \right\rceil - 1. \tag{A.7}$$

and the possible positive HR tick difference (largest value of $\hat{t}_{\mathrm{TM},k}$*, largest value of* Δc_{Si} *in* (A.5) *and smallest value of* $\hat{N}_{\mathrm{Si},k}$*) is*

$$\Delta N_{\mathrm{Si}}^{\max} = \left\lceil \frac{t_{\mathrm{unc}} \cdot d_{\mathrm{Si}}}{T_{\mathrm{Si}}} \right\rceil + \left\lceil \frac{J_{\mathrm{TM}}}{T_{\mathrm{Si}}} \right\rceil + \left\lceil \frac{E_{\mathrm{TRI}}}{T_{\mathrm{Si}}/(1+d_{\mathrm{Si}})} \right\rceil \tag{A.8}$$

That is, indeed

$$|\Delta c_{\mathrm{Si}}(t_k^+)| \leq \max\{|\Delta N_{\mathrm{Si}}^{\min}|, |\Delta N_{\mathrm{Si}}^{\max}|\} \cdot T_{\mathrm{Si}}.$$

This confirms (A.2)*. An upper bound for the clock difference between the LCs of TM and Si is then given by adding the maximum clock difference after offset corrections and the accumulated clock difference between offset corrections due to clock drift in the form*

$$\begin{aligned} \max_t\{|\Delta c_{\mathrm{Si}}(t)|\} \leq {} & \max\{|\Delta N_{\mathrm{Si}}^{\min}|, |\Delta N_{\mathrm{Si}}^{\max}|\} \cdot T_{\mathrm{Si}} \\ & + \left\lceil \frac{|d_{\mathrm{Si}}| \cdot T}{T_{\mathrm{Si}}} \right\rceil \cdot T_{\mathrm{Si}}. \end{aligned}$$

Appendix B

PROOFS IN CHAPTER 6

B.1 Proof of Theorem 2

We next provide the proof of Theorem 2. We restate the theorem for convenience.

Theorem 2 *Consider two natural numbers* $N, M \in \mathbb{N}$ *such that* $N = n \cdot g + \Delta n$ *and* $M = m \cdot g + \Delta m$ *for* $m, n, g \in \mathbb{N}$*,* m, n *coprime and* $\Delta n, \Delta m \in \mathbb{Z}$*. Further assume that there are bounds* $X, Y \in \mathbb{N}$ *such that* $|\Delta n|, |\Delta m| < X$ *and* $g > Y$*. Then, it holds that* m, n *are the unique coprime natural numbers such that*

$$|N \cdot m - M \cdot n| < X \cdot (m + n). \tag{B.1}$$

and

$$2 \cdot X \cdot (m + n) < Y. \tag{B.2}$$

In addition consider any estimate $\hat{g} = \left\lfloor \lambda \cdot \frac{N}{n} + (1 - \lambda) \cdot \frac{M}{m} \right\rceil$ *of g for* $0 \leq \lambda \leq 1$*. Then,*

$$|g - \hat{g}| < \left\lfloor \frac{X}{\min\{n, m\}} \right\rceil. \tag{B.3}$$

Proof 2 *Without loss of generality, assume that* $N > M$ *such that* $n \geq m$*. Further, assume that* (B.1) *and* (B.2) *are fulfilled. That is,*

$$|N \cdot m - M \cdot n| < X \cdot (m + n) \text{ and } 2 \cdot X \cdot (m + n) < Y.$$

In order to show that m, n *are unique, we pursue a proof by contradiction. To this end, we assume that there are different coprime numbers* $k, l \in \mathbb{N}$ *with* $k \neq n$ *and/or* $l \neq m$ *such that*

$$|N \cdot l - M \cdot k| < X \cdot (k + l) \text{ and } 2 \cdot X \cdot (k + l) < Y.$$

Then, we have

$$\begin{aligned}|N \cdot l - M \cdot k| &= |(n \cdot g + \Delta n) \cdot l - (m \cdot g + \Delta m) \cdot k| \\ &= |(n \cdot l - m \cdot k) \cdot g + \Delta n \cdot l - \Delta m \cdot k| \\ &< (l + k) \cdot X.\end{aligned}$$

There are two cases. If $n \cdot l - m \cdot k = 0$*, it holds that*

$$\frac{n}{m} = \frac{k}{l}.$$

Since k, l *are coprime and* $k \neq n$ *and/or* $l \neq m$ *by assumption, this implies that* $n = c \cdot k$ *and* $m = c \cdot l$ *for some* $c > 1$*. Hence,* n *and* m *are not coprime, which violates the assumption. If* $n \cdot l - m \cdot k \neq 0$*, we conclude that* $|(n \cdot l - m \cdot k) \cdot g| \geq g$*. Since* $g > Y$ *by assumption, we compute*

$$\begin{aligned}|N \cdot l - M \cdot k| &= |(n \cdot l - m \cdot k) \cdot g + \Delta n \cdot l - \Delta m \cdot k| \\ &\geq |(n \cdot l - m \cdot k) \cdot g| - |\Delta n \cdot l - \Delta m \cdot k| \\ &> g - (k + l) \cdot X > Y - (k + l) \cdot X \\ &> 2 \cdot (k + l) \cdot X - (k + l) \cdot X \\ &> (k + l) \cdot X.\end{aligned}$$

This contradicts the assumption that $|N \cdot l - M \cdot k| < X \cdot (k + l)$*. Hence,* m *and* n *are the unique coprime natural numbers that fulfill* (B.1) *and* (B.2)*.*

We next consider the estimate $\hat{g} = \left\lfloor \lambda \cdot \frac{N}{n} + (1 - \lambda) \cdot \frac{M}{m} \right\rceil$ *with* $0 \leq \lambda \leq 1$*. It holds that*

$$\begin{aligned}|g - \hat{g}| &= \left| g - \left\lfloor \lambda \cdot \frac{N}{n} + (1 - \lambda) \cdot \frac{M}{m} \right\rceil \right| \\ &= \left| g - \left\lfloor \lambda \cdot \frac{n \cdot g + \Delta n}{n} + (1 - \lambda) \cdot \frac{m \cdot g + \Delta m}{m} \right\rceil \right| \\ &= \left| g - \left\lfloor \lambda \cdot g + (1 - \lambda) \cdot g + \frac{\lambda \cdot \Delta n}{n} + \frac{(1 - \lambda) \cdot \Delta m}{m} \right\rceil \right| \\ &= \left| \left\lfloor \frac{\lambda \cdot \Delta n}{n} + \frac{(1 - \lambda) \cdot \Delta m}{m} \right\rceil \right| \leq \left| \left\lfloor (\frac{\lambda}{n} + \frac{1 - \lambda}{m}) \cdot X \right\rceil \right| \\ &\leq \left| \left\lfloor \frac{X}{\min\{n, m\}} \right\rceil \right|.\end{aligned}$$

This concludes the proof of Theorem 2.

B.2 Proof of Theorem 3

We next prove Theorem 3 which is also restated for convenience.

Theorem 3 *(i) It holds at each iteration j of Algorithm 7 with measurement $\Delta q_j = n_j \cdot P_{\text{Si}} + \Delta p_j$ and multiple $\overline{n}_j$ in line (11) that $\overline{P}_{\text{Si},j} = \overline{k}_j \cdot P_{\text{Si}} + \Delta\overline{p}_j$ with $\overline{k}_j = \min_{r=1,\dots,j}\{\frac{n_r}{\overline{n}_r}\}$ and $|\Delta\overline{p}_j| < X$. (ii) If additionally $\overline{n}_j = n_j$ in line 11 of Algorithm 7 and there is no zero entry in v, it holds for all $r \geq j$ that*

$$|\overline{P}_{\text{Si},r} - P_{\text{Si}}| < X_{\text{eff}} = X \cdot \frac{H_L}{L}. \tag{B.4}$$

Hereby, $H_L = \sum_{n=1}^{L} \frac{1}{n}$ denotes the L-th harmonic number.

Proof 3 *We prove the first part of the theorem by induction. For the initialization, we note that $\overline{P}_{\text{Si},1} = \Delta q_1 = n_1 \cdot P_{\text{Si}} + \Delta p_1$ in line 9. Using $\overline{n}_1 = 1$, $\overline{k}_1 = n_1 = \min_{r=1}\{n_r/\overline{n}_r\}$ and $\Delta\overline{p}_1 = \Delta p_1$, it follows that $|\Delta\overline{p}_1| < X$ by assumption. Moreover, only $v[1,1] = \Delta q_1 \neq 0$ by line 3 and 9 with $|v[1,1] - \overline{k}_1 \cdot P_{\text{Si}}| = |n_1 \cdot P_{\text{Si}} + \Delta p_1 - n_1 \cdot P_{\text{Si}}| = |\Delta p_1| < X$.*

For the induction step, we assume for some iteration j that $\overline{P}_{\text{Si},j} = \overline{k}_j \cdot P_{\text{Si}} + \Delta\overline{p}_j$ with $|\Delta\overline{p}_j| < X$, $\overline{k}_j = \min_{r=1,\dots,j}\{n_r/\overline{n}_r\}$ and $|v[l,s] - \overline{k}_j \cdot P_{\text{Si}}| < X$ for all $l \leq L$ and $s \leq S$ such that $v[l,s] \neq 0$. We next look at iteration $j+1$. If there are no integers $\overline{m}_{j+1}, \overline{n}_{j+1}$ to fulfill the condition in line 11, it holds that v and c are unchanged and $\overline{P}_{\text{Si},j+1} = \overline{P}_{\text{Si}}, j$ with line 21. That is, v and $\overline{P}_{\text{Si},j+1}$ trivially fulfill the assertion. Otherwise, we have that $\Delta q_{j+1} = n_{j+1} \cdot P_{\text{Si}} + \Delta p_{j+1}$ with

$$\begin{aligned}
&|\overline{n}_{j+1} \cdot \overline{P}_{\text{Si},j} - \overline{m}_{j+1} \cdot \Delta q_{j+1}| = \\
&\quad = |\overline{n}_{j+1} \cdot \overline{k}_j \cdot P_{\text{Si}} + \overline{n}_{j+1} \cdot \Delta\overline{p}_j \\
&\qquad\quad - \overline{m}_{j+1} \cdot n_{j+1} \cdot P_{\text{Si}} - \overline{m}_{j+1} \cdot \Delta p_{j+1}| \\
&\quad = |(\overline{n}_{j+1} \cdot \overline{k}_j - \overline{m}_{j+1} \cdot n_{j+1}| \cdot P_{\text{Si}} \\
&\qquad\quad + \overline{n}_{j+1} \cdot \Delta\overline{p}_j - \overline{m}_{j+1} \cdot \Delta p_{j+1}| \\
&\quad < (\overline{m}_{j+1} + \overline{n}_{j+1}) \cdot X
\end{aligned}$$

because of line 11 in Algorithm 7. In analogy to the proof of Theorem 2, this implies that $\overline{n}_{j+1} \cdot \overline{k}_j - \overline{m}_{j+1} \cdot n_{j+1} = 0$. Now we consider two cases.

(i) In the first case, $\overline{m}_{j+1} = 1$. Then, it holds that v does not change before line 19 in Algorithm 7. Moreover,

$$\begin{aligned} v[\overline{n}_{j+1}, c[\overline{n}_{j+1}]] &= \lfloor \frac{\Delta q_{j+1}}{\overline{n}_{j+1}} \rceil = \lfloor \frac{n_{j+1} \cdot P_{\mathrm{Si}} + \Delta p_{j+1}}{\overline{n}_{j+1}} \rceil \\ &= \lfloor \frac{\overline{n}_{j+1} \cdot \overline{k}_j}{\overline{n}_{j+1}} \cdot P_{\mathrm{Si}} + \frac{\Delta p_{j+1}}{\overline{n}_{j+1}} \rceil = \overline{k}_j \cdot P_{\mathrm{Si}} + \lfloor \frac{\Delta p_{j+1}}{\overline{n}_{j+1}} \rceil \end{aligned}$$

with $|\lfloor \frac{\Delta p_{j+1}}{\overline{n}_{j+1}} \rceil| < X$ since $\overline{n}_{j+1} \geq 1$. Hence, using $\overline{k}_{j+1} = \overline{k}_j = \frac{n_{j+1}}{\overline{n}_{j+1}}$, we have that $\overline{k}_{j+1} = \min_{r=1,\ldots,j+1}\{n_r/\overline{n}_{r+1}\}$ and for all $l \leq L$ and $s \leq S$ that $|v[l,s] - \overline{k}_{j+1} \cdot P_{\mathrm{Si}}| < X$ if $v[l,s] \neq 0$. Further, $\overline{P}_{\mathrm{Si},j+1} = \overline{k}_{j+1} \cdot P_{\mathrm{Si}} + \Delta\overline{p}_{j+1}$ with $|\Delta\overline{p}_{j+1}| < X$ after taking the average in line 24.

(ii) In the second case, $\overline{m}_{j+1} > 1$. Then, $\overline{n}_{j+1} \cdot \overline{k}_j - \overline{m}_{j+1} \cdot n_{j+1} = 0$ implies that $n_{j+1} = \frac{\overline{n}_{j+1}}{\overline{m}_{j+1}} \cdot \overline{k}_j$. Since $n_{j+1} \in \mathbb{N}$ and $\overline{m}_{j+1}, \overline{n}_{j+1}$ are coprime it follows that $\overline{k}_j = k \cdot \overline{m}_{j+1}$ for some $k \in \mathbb{N}$. That is, $n_{j+1} = \frac{\overline{n}_{j+1} \cdot k \cdot \overline{m}_{j+1}}{\overline{m}_{j+1}} = \overline{n}_{j+1} \cdot k$. According to line 14 and 15 in Algorithm 7, we update $v[l \cdot \overline{m}_{j+1}, s] = \lfloor v[l,s]/\overline{m}_{j+1} \rceil$. Choosing $\overline{k}_{j+1} = \overline{k}_j/\overline{m}_{j+1} = (k \cdot \overline{m}_{j+1})/\overline{m}_{j+1} = k$, it follows that $\overline{k}_{j+1} = \min_{r=1,\ldots,j+1}\{n_r/\overline{n}_{r+1}\}$ and

$$\begin{aligned} |v[l \cdot \overline{m}_{j+1}, s] - \overline{k}_{j+1} \cdot P_{\mathrm{Si}}| &= |\frac{v[l,s] - \overline{k}_j \cdot P_{\mathrm{Si}}}{\overline{m}_{j+1}}| \\ &= \frac{1}{\overline{m}_{j+1}} \cdot |v[l,s] - \overline{k}_j \cdot P_{\mathrm{Si}}| < \frac{X}{\overline{m}_{j+1}} < X \end{aligned}$$

for all $l \leq L$ and $s \leq S$ such that $v[l,s] \neq 0$. In addition,

$$\begin{aligned} v[\overline{n}_{j+1}, c[\overline{n}_{j+1}]] &= \lfloor \frac{\Delta q_{j+1}}{\overline{n}_{j+1}} \rceil = \lfloor \frac{n_{j+1} \cdot P_{\mathrm{Si}} + \Delta p_{j+1}}{\overline{n}_{j+1}} \rceil \\ &= \lfloor \frac{\overline{n}_{j+1} \cdot \overline{k}_j \cdot P_{\mathrm{Si}}}{\overline{n}_{j+1} \cdot \overline{m}_{j+1}} + \frac{\Delta p_{j+1}}{\overline{n}_{j+1}} \rceil \\ &= k \cdot P_{\mathrm{Si}} + \lfloor \frac{\Delta p_{j+1}}{\overline{n}_{j+1}} \rceil = \overline{k}_{j+1} \cdot P_{\mathrm{Si}} + \lfloor \frac{\Delta p_{j+1}}{\overline{n}_{j+1}} \rceil \end{aligned}$$

with $|\lfloor \frac{\Delta p_{j+1}}{\overline{n}_{j+1}} \rceil| < X$ since $\overline{n}_{j+1} \geq 1$. Hence, we conclude that $|v[\overline{n}_{j+1}, c[\overline{n}_{j+1}]] - \overline{k}_{j+1} \cdot P_{\mathrm{Si}} < X$. Further, we know that $\overline{P}_{\mathrm{Si},j+1} = \overline{k}_{j+1} \cdot P_{\mathrm{Si}} + \Delta\overline{p}_{j+1}$ with $|\Delta\overline{p}_{j+1}| < X$ when taking the average in line 24.

Together, $\overline{P}_{\mathrm{Si},j} = \overline{k}_j \cdot P_{\mathrm{Si}} + \Delta\overline{p}_j$ with $\overline{k}_j = \min_{r=1,\ldots,j}\{\frac{n_r}{\overline{n}_r}\}$ and $|\Delta\overline{p}_j| < X$ in each iteration j.

It remains to show that

$$|\overline{P}_{\mathrm{Si},j} - P_{\mathrm{Si}}| < X \cdot \frac{H_{L-1}}{L}$$

under the assumption that $\overline{n}_j = n_j$ *in line 11 of Algorithm 7 and there is no zero entry in* v*. To this end, we assume that* $\overline{n}_j = n_j$ *in line 11 of Algorithm 7 and there is no zero entry in* v*. Then, it follows from the previous part of the proof that* $\overline{k}_j = \min_{r=1,\dots,j}\{\frac{n_r}{\overline{n}_j}\} = 1$*. That is,* $\overline{P}_{\mathrm{Si},j} = P_{\mathrm{Si}} + \Delta\overline{p}_i$*. We next analyze the error for each multiple* l *of* P_{Si}*. To this end, we recall that* $\Delta q_j/l$ *is inserted in row* l *of* v*. Since* $\Delta q_j = l \cdot P_{\mathrm{Si}} + \Delta p_i$ *and* $|\Delta p_i| < X$*, it must hold that*

$$|\frac{\Delta q_j}{l} - P_{\mathrm{Si}}| = |\frac{\Delta p_i}{l}| \leq \frac{X}{l}.$$

Considering that $\overline{P}_{\mathrm{Si},j}$ *is the average of the non-zero values in* v_i*, it directly follows that*

$$\begin{aligned} |\overline{P}_{\mathrm{Si},j} - P_{\mathrm{Si}}| &= \frac{1}{L \cdot S} \cdot |\sum_{l=1}^{L}\sum_{s=1}^{S} v[l,s] - P_{\mathrm{Si}}| \\ &\leq \frac{1}{L \cdot S} \cdot \sum_{l=1}^{L}\sum_{s=1}^{S} \frac{X}{l} = \frac{X}{L} \cdot \sum_{l=1}^{L} \frac{1}{l} = X \cdot \frac{H_L}{L}. \end{aligned}$$

Considering that $\overline{k}_j \geq 1$*, it also follows that* $\overline{k}_r = 1$ *for all* $r \geq j$*. Hence, indeed,* $|\overline{P}_{\mathrm{Si},} - P_{\mathrm{Si}}| < X \cdot \frac{H_L}{L}$ *for all* $r \geq j$*.*

B.3 Clock Accuracy of ACS-PEDC

We next derive the bound on the clock accuracy in (6.8) of ACS-PEDC. To this end, we consider an arbitrary cycle k between the times t_k and t_{k+1}.

ACS-PEDC applies drift correction between the offset corrections after the RMs at t_k and t_{k+1}, that is, during a time duration T. Considering the perfect correction period P_{Si} in Section 6.1 in the main document, the required number of clock corrections is given by

$$\left\lfloor \frac{T}{\gamma \cdot (P_{\mathrm{Si}} - s_{\mathrm{Si}}) \cdot T_{\mathrm{Si}}} \right\rfloor = \left\lfloor \frac{T \cdot (1 + d_{\mathrm{Si}})}{\gamma \cdot T_{\mathrm{Si}} \cdot P_{\mathrm{Si}}} \right\rfloor. \tag{B.5}$$

In contrast, the number of actually applied clock corrections in iteration j of Algorithm 7 is

$$\left\lfloor \frac{T \cdot (1 + d_{\mathrm{Si}})}{\gamma \cdot T_{\mathrm{Si}} \cdot \overline{P}_{\mathrm{Si},j}} \right\rfloor \tag{B.6}$$

with the estimated correction period $\overline{P}_{\mathrm{Si},j}$.

We next evaluate the difference between the required number of clock corrections with period P_{Si} and the number of actually applied clock corrections with period $\overline{P}_{\mathrm{Si},j}$ as

$$
\begin{aligned}
&\Big|\Big\lfloor\frac{T\cdot(1+d_{\mathrm{Si}})}{\gamma\cdot T_{\mathrm{Si}}\cdot P_{\mathrm{Si}}}\Big\rfloor-\Big\lfloor\frac{T\cdot(1+d_{\mathrm{Si}})}{\gamma\cdot T_{\mathrm{Si}}\cdot \overline{P}_{\mathrm{Si},j}}\Big\rfloor\Big|\\
=&\Big|\Big\lfloor\frac{T\cdot(1+\frac{s_{\mathrm{Si}}}{P_{\mathrm{Si}}-s_{\mathrm{Si}}})}{\gamma\cdot T_{\mathrm{Si}}\cdot P_{\mathrm{Si}}}\Big\rfloor-\Big\lfloor\frac{T\cdot(1+\frac{s_{\mathrm{Si}}}{P_{\mathrm{Si}}-s_{\mathrm{Si}}})}{\gamma\cdot T_{\mathrm{Si}}\cdot \overline{P}_{\mathrm{Si},j}}\Big\rfloor\Big|\\
=&\Big|\Big\lfloor\frac{T}{\gamma\cdot T_{\mathrm{Si}}\cdot(P_{\mathrm{Si}}-s_{\mathrm{Si}})}\Big\rfloor-\Big\lfloor\frac{T\cdot P_{\mathrm{Si}}}{\gamma\cdot T_{\mathrm{Si}}\cdot \overline{P}_{\mathrm{Si},j}\cdot(P_{\mathrm{Si}}-s_{\mathrm{Si}})}\Big\rfloor\Big|
\end{aligned}
$$

using the relation between d_{Si} and P_{Si} in (5.5) in the main document. We next consider the general relation

$$
\lfloor x\rfloor-\lfloor y\rfloor\le\lceil x-y\rceil, \tag{B.7}
$$

which holds for any real numbers $x,y\in\mathbb{R}$. We further recall that $|\overline{P}_{\mathrm{Si},j}-P_{\mathrm{Si}}|<X_{\mathrm{eff}}$ according to Theorem 3. Hence, we can continue

$$
\begin{aligned}
&\Big|\Big\lfloor\frac{T\cdot(1+d_{\mathrm{Si}})}{\gamma\cdot T_{\mathrm{Si}}\cdot P_{\mathrm{Si}}}\Big\rfloor-\Big\lfloor\frac{T\cdot(1+d_{\mathrm{Si}})}{\gamma\cdot T_{\mathrm{Si}}\cdot \overline{P}_{\mathrm{Si},j}}\Big\rfloor\Big|\\
\le&\Big|\Big\lceil\frac{T}{\gamma\cdot T_{\mathrm{Si}}\cdot(P_{\mathrm{Si}}-s_{\mathrm{Si}})}-\frac{T\cdot P_{\mathrm{Si}}}{\gamma\cdot T_{\mathrm{Si}}\cdot \overline{P}_{\mathrm{Si},j}\cdot(P_{\mathrm{Si}}-s_{\mathrm{Si}})}\Big\rceil\Big|\\
=&\Big|\Big\lceil\frac{T\cdot(\overline{P}_{\mathrm{Si},j}-P_{\mathrm{Si}})}{\gamma\cdot T_{\mathrm{Si}}\cdot \overline{P}_{\mathrm{Si},j}\cdot(P_{\mathrm{Si}}-s_{\mathrm{Si}})}\Big\rceil\Big|\\
\le&\Big|\Big\lceil\frac{T\cdot X_{\mathrm{eff}}}{\gamma\cdot T_{\mathrm{Si}}\cdot(P_{\mathrm{Si}}-X_{\mathrm{eff}})\cdot(P_{\mathrm{Si}}-1)}\Big\rceil\Big|.
\end{aligned}
$$

In addition, a clock difference of at most γ HR ticks is accumulated between the clock corrections with period $\overline{P}_{\mathrm{Si},j}$ and after the last clock correction before the next RM at t_{k+1}. Adding the maximum absolute clock difference after offset corrections and considering that each correction corresponds to γ HR ticks, the clock accuracy of ACS-PEDC is indeed determined by

$$
\begin{aligned}
&\max_t\{|c_{\mathrm{Si}}(t)|\}\le C_{\mathrm{OC}}+\\
&+\Big(\Big|\Big\lceil\frac{T\cdot X_{\mathrm{eff}}}{\gamma\cdot T_{\mathrm{Si}}\cdot(P_{\mathrm{Si}}-X_{\mathrm{eff}})\cdot(P_{\mathrm{Si}}-1)}\Big\rceil\Big|+1\Big)\cdot\gamma\cdot T_{\mathrm{Si}}.
\end{aligned}
\tag{B.8}
$$

B.4 Start-up algorithm

This section provides the detailed derivations of the drift estimates and bounds in Section 6.5.1 of the main document. We first consider the drift estimate in (6.10). Ideally, it holds that $(\hat{N}_{\mathrm{Si},1} - \lfloor \hat{t}_{\mathrm{TM},1}/T_{\mathrm{Si}} \rfloor) \cdot T_{\mathrm{Si}} = 0$. That is, the clock difference is zero with the offset correction after the first RM. Assuming no further offset correction and drift correction after the first RM, it is the case that $(\hat{N}_{\mathrm{Si},k} - \lfloor \hat{t}_{\mathrm{TM},k}/T_{\mathrm{Si}} \rfloor) \cdot T_{\mathrm{Si}}$ represents the accumulated clock difference between TM and Si after the k-th RM, that is, within the time duration $(k-1) \cdot T$. Assuming a constant drift between the first and k-th RM, it indeed follows from (2.5) that

$$d^{\mathrm{RM}}_{\mathrm{Si},k} = \frac{(\hat{N}_{\mathrm{Si},k} - \lfloor \hat{t}_{\mathrm{TM},k}/T_{\mathrm{Si}} \rfloor) \cdot T_{\mathrm{Si}}}{(k-1) \cdot T}, \tag{B.9}$$

which confirms (6.10) in the main document. We next determine the clock correction error $E^{\mathrm{CC}}_{\mathrm{Si}}$ in (6.11) in the main document. $\Delta N^{\min}_{\mathrm{Si}}$ in (A.7) and $\Delta N^{\max}_{\mathrm{Si}}$ in (A.8) in Appendix A.1 represent the possible minimum and maximum HR tick difference after offset correction. That is, the maximum HR tick error of $\hat{N}_{\mathrm{Si},k} - \lfloor \hat{t}_{\mathrm{TM},k}/T_{\mathrm{Si}} \rfloor$ in (B.9) occurs if the HR tick difference is $\Delta N^{\min}_{\mathrm{Si}}$ after RM 1 and $\Delta N^{\max}_{\mathrm{Si}}$ after RM k or $\Delta N^{\max}_{\mathrm{Si}}$ after RM 1 and $\Delta N^{\min}_{\mathrm{Si}}$ after RM k. In both cases, we obtain the maximum absolute clock correction error as

$$\begin{aligned} E^{\mathrm{CC}}_{\mathrm{Si}} &= \left|\Delta N^{\max}_{\mathrm{Si}} - \Delta N^{\min}_{\mathrm{Si}}\right| \cdot T_{\mathrm{Si}} = \\ &= \left(\left\lceil \frac{J_{\mathrm{TM}}}{T_{\mathrm{Si}}} \right\rceil + \left\lceil \frac{E_{\mathrm{TRI}}}{T_{\mathrm{Si}}/(1+d_{\mathrm{Si}})} \right\rceil + \left\lceil \frac{E_{\mathrm{TRI}} + J_{\mathrm{Si}}}{T_{\mathrm{Si}}/(1+d_{\mathrm{Si}})} \right\rceil + 1\right) \cdot T_{\mathrm{Si}}. \end{aligned} \tag{B.10}$$

Using (B.9) and (B.10), we determine the lower and upper bounds $d^{\mathrm{lo}}_{\mathrm{Si},k}$ and $d^{\mathrm{up}}_{\mathrm{Si},k}$ for the actual drift as

$$\begin{aligned} d^{\mathrm{lo}}_{\mathrm{Si},k} &= \frac{(\hat{N}_{\mathrm{Si},k} - \lfloor \hat{t}_{\mathrm{TM},k}/T_{\mathrm{Si}} \rfloor) \cdot T_{\mathrm{Si}} - E^{\mathrm{CC}}_{\mathrm{Si}}}{(k-1) \cdot T} \\ &= d^{\mathrm{RM}}_{\mathrm{Si},k} - \frac{E^{\mathrm{CC}}_{\mathrm{Si}}}{(k-1) \cdot T}, \end{aligned} \tag{B.11}$$

$$\begin{aligned} d^{\mathrm{up}}_{\mathrm{Si},k} &= \frac{(\hat{N}_{\mathrm{Si},k} - \lfloor \hat{t}_{\mathrm{TM},k}/T_{\mathrm{Si}} \rfloor) \cdot T_{\mathrm{Si}} + E^{\mathrm{CC}}_{\mathrm{Si}}}{(k-1) \cdot T} \\ &= d^{\mathrm{RM}}_{\mathrm{Si},k} + \frac{E^{\mathrm{CC}}_{\mathrm{Si}}}{(k-1) \cdot T}. \end{aligned} \tag{B.12}$$

This confirms (6.12) in the main document.

We finally provide the proof of Theorem 4 in the main document, which is restated here for convenience.

Theorem 4 *Consider Algorithm 8 and write* t_{ST} *for the start-up time, when* `active` = `ACS-PEDC`*. Then,*

$$\max_t\{|c_{\mathrm{Si}}(t)|\} \le \begin{cases} \Delta_{\mathrm{RM}} = C_{\mathrm{OC}} + \frac{E_{\mathrm{Si}}^{\mathrm{CC}}}{(k-1)} & \text{if } T \le t \le t_{\mathrm{ST}} \\ \Delta_{\mathrm{PEO}} \text{ in } (6.8) & \text{if } t > t_{\mathrm{ST}}. \end{cases} \tag{B.13}$$

Proof 4 *Initially,* `active` $=$ *START-UP according to Algorithm 8. In each iteration of the while loop in line 2, it is first checked if at least two RMs were received. Assuming that the first RM is transmitted at time 0, two RMs are received at time* T*. Before time* T*, no guarantees on the clock difference are given. After two RMs are received, it is checked if there is a unique value of* m *in* (6.15)*.* `active` $=$ *START-UP as long as no unique* m *is determined. Specifically,* $t \le t_{\mathrm{ST}}$ *as long as* `active` $=$ *START-UP. In that case, it holds that the applied drift correction is* $d_{\mathrm{Si},k}^{\mathrm{RM}}$*, whereas the actual drift is given by the lower and upper bounds in* (6.12)*. That is, the maximum accumulated clock difference because of an inaccurate drift estimate is given by*

$$\begin{aligned}\left(d_{\mathrm{Si},k}^{\mathrm{up}} - d_{\mathrm{Si},k}^{\mathrm{RM}}\right) \cdot T &= \left(d_{\mathrm{Si},k}^{\mathrm{RM}} - d_{\mathrm{Si},k}^{\mathrm{RM}}\right) \cdot T \\ &= \frac{E_{\mathrm{Si}}^{\mathrm{CC}}}{(k-1)\cdot T} \cdot T = \frac{E_{\mathrm{Si}}^{\mathrm{CC}}}{(k-1)}.\end{aligned}$$

Considering the bound on the clock difference from the offset correction in (5.3) *in the main document, it holds for* $t \le t_{\mathrm{ST}}$ *that*

$$\max_t\{c_{\mathrm{Si}}(t)\} \le C_{\mathrm{OC}} + \frac{E_{\mathrm{Si}}^{\mathrm{CC}}}{(k-1)},$$

which confirms (6.16) *in the main document. If a unique value of* m *is found in* (6.15)*, Algorithm 8 sets* `active` $=$ *ACS-PEDC and* $t > t_{\mathrm{ST}}$*. Then, we know that*

$$\overline{P}_{\mathrm{Si},j} = m \cdot P_{\mathrm{Si}} + \Delta\overline{p}_j$$

such that

$$\left|\frac{\overline{P}_{\mathrm{Si},j}}{m} - P_{\mathrm{Si}}\right| = \left|\frac{m \cdot P_{\mathrm{Si}} + \Delta\overline{p}_j}{m} - P_{\mathrm{Si}}\right| \le \frac{X_{\mathrm{eff}}}{m} \le X_{\mathrm{eff}}$$

since $m \ge 1$*. But then, the bound on the clock difference in* (6.8) *in the main document holds, which concludes the proof.*

Appendix C

PROOFS IN CHAPTER 7

C.1 Proof of Theorem 5

We prove Theorem 5 from the main document, which is recalled here for convenience.

Theorem 5 *Consider a feasible WTDMA schedule as in Definition 1 with the parameters introduced above. Further, define* $w_{\min} = \min_{M_i \in \mathcal{M}} L_i^{\max}$ *as the minimum window size for the messages in* $\mathcal{M}$ *and* $\Delta_{\max} = \max_{N \in \mathcal{N}} \Delta_N$ *as the maximum software delay of the nodes in* $\mathcal{N}$*. Then,* $V = 0$ *if*

$$w_{\min} > w_{\text{safe}} = \Delta_{\max} + 2 \cdot \theta. \tag{C.1}$$

Hereby, w_{safe} *denotes the safe window size. Furthermore, the starting time delay* $\phi_{j,b}$ *and the response time* $R_{j,b}$ *of any message instance* $M_{j,b}$ *are bounded by*

$$\phi_{j,b} \leq \phi^{\max} = \Delta_{\max} + 2\,\theta \text{ and } R_{j,b} \leq R_j^{\max} = \phi^{\max} + L_j^{\max}. \tag{C.2}$$

Proof 5 *We first consider a generic window* (j, b) *for the b-th instance* $M_{j,b}$ *of message* $M_j \in \mathcal{M}$ *and show that (1)* $y_{j,b} - \theta \leq s_{j,b} \leq y_{j,b} + \Delta_{\max} + \theta$ *and (2) all messages scheduled earlier than* $M_{j,b}$ *complete their transmission no later than* $y_{j,b} + \Delta_{\max} + \theta$ *by induction. For the induction basis, consider the window* $(j, b) = (j, 0)$*, assuming that instance* $M_{j,0}$ *is transmitted in the first window of the schedule. Regarding* $M_{j,0}$*, it is known that* $y_{j,0} - \theta \leq e_{j,0} \leq y_{j,0} + \theta + \Delta_{\max}$ *with F1. Besides, the CAN bus is idle until* $M_{j,0}$ *starts transmission and subsequent messages in the schedule can not be ready before* $M_{j,0}$ *due to* (C.1)*. Thus, it holds that (1)* $y_{j,0} - \theta \leq s_{j,0} \leq y_{j,0} + \Delta_{\max} + \theta$ *and (2) all messages which are scheduled earlier than* $M_{j,0}$ *complete their transmission no later than* $y_{j,0} + \Delta_{\max} + \theta$*. Now consider two consecutive windows* (i, a)

and (j, b) *with the induction assumption (1)* $y_{i,a} - \theta \leq s_{i,a} \leq y_{i,a} + \Delta_{\max} + \theta$ *and (2) all messages which are scheduled earlier than* $M_{i,a}$ *complete their transmission no later than* $y_{i,a} + \Delta_{\max} + \theta$*. Hence, transmission of* $M_{i,a}$ *finishes at* $f_{i,a} \leq y_{i,a} + \Delta_{\max} + \theta + L_{i,a} \leq y_{i,a} + \Delta_{\max} + \theta + w_i \leq y_{j,b} + \Delta_{\max} + \theta$ *considering F3. That is, it follows that all messages scheduled earlier than* $M_{j,b}$ *complete their transmission no later than* $y_{j,b} + \Delta_{\max} + \theta$*. Since* $M_{j,b}$ *is ready at* $e_{j,b}$ *such that* $y_{j,b} - \theta \leq e_{j,b} \leq y_{j,b} + \Delta_{\max} + \theta$ *with F1 and subsequent messages in the schedule can not be ready before* $M_{j,b}$ *due to* (C.1)*, it holds that* $y_{j,b} - \theta \leq s_{j,b} \leq y_{j,b} + \Delta_{\max} + \theta$*. This concludes the induction step.*

Based on this result, it is now possible to prove that $V = 0$ *when using WTDMA. Formally, considering any arbitrary two consecutive message instances* $M_{i,a}$ *and* $M_{j,b}$*, it must hold that* $s_{i,a} < s_{j,b}$*. Since the WTDMA schedule is feasible, it holds that* $y_{i,a} + \Delta_{\max} + \theta < y_{j,b} - \theta$ *due to* (C.1)*. Furthermore, we know from the above computation that* $s_{i,a} \leq y_{i,a} + \Delta_{\max} + \theta$ *and* $y_{j,b} - \theta \leq s_{j,b}$*. This directly implies that* $s_{i,a} < s_{j,b}$ *with* $y_{i,a} + \Delta_{\max} + \theta < y_{j,b} - \theta$*. Since* $s_{i,a} < s_{j,b}$ *and* i, j, a, b *were chosen arbitrarily, it follows that indeed* $V = 0$*.*

Regarding the latest starting time, we use $s_{j,b} \leq y_{j,b} + \Delta_{\max} + \theta$ *from above. Since* $g_{j,b} \geq y_{j,b} - \theta$*, it follows that* $s_{j,b} - g_{j,b} \leq \Delta_{\max} + 2\,\theta = \phi^{\max}$*. Adding the longest transmission duration of instance* $M_{j,b}$ *to* $\phi_{\max}$*, we obtain the WCRT* $R_{j,b} \leq R_j^{\max} = \Delta_{\max} + 2\,\theta + L_j^{\max}$*.*

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Akpınar, Murat
Nationality: Turkish (TC)
Date and Place of Birth: 21.12.1988, İzmir,Turkey
Marital Status: Married
Phone: +905058151763

EDUCATION

Degree	**Institution**	**Year of Graduation**
M.S.	EEE Dept., METU	2014
B.S.	EEE Dept., METU	2011
High School	Aydın Science High School, Aydın	2006

PROFESSIONAL EXPERIENCE

Year	**Place**	**Enrollment**
June 2011 - Present	Aselsan Inc., Ankara	Digital Design Engineer

PUBLICATIONS

International Journals

- M. Akpınar, K. W. Schmidt, and E. G. Schmidt, "Highly Accurate Clock Synchronization with Drift Correction for the Controller Area Network," IEEE

Transactions on Parallel and Distributed Systems, Accepted

- M. Akpınar, K. W. Schmidt, and E. G. Schmidt, "Weak TDMA for the Deterministic Medium Access on the Controller Area Network," IEEE Transactions on Intelligent Transportation Systems, Under Review.
- M. Akpınar and K. W. Schmidt, "Predictable Timestamping for the Controller Area Network Evaluation and Effect on Clock Synchronization Accuracy," IEEE Transactions on Systems, Man and Cybernetics: Systems, Under Review.

International Conference Publications

- M. Akpınar, K. W. Schmidt, and E. G. Schmidt, “Improved clock synchronization algorithms for the controller area network (CAN),” in International Conference on Computer Communication and Networks, pp. 1–8, IEEE, 2019.
- M. Akpınar, E. G. Schmidt, and K. Werner Schmidt, “Drift correction for the software-based clock synchronization on controller area network,” in 2020 IEEE Symposium on Computers and Communications (ISCC), pp. 1–6, 2020.

National Conference Publications

- M. Akpınar, E. G. Schmidt, and K. W. Schmidt, “Evaluation of clock synchronization algorithms for controller area network,” in Signal Processing and Communications Applications Conference, May 2020, pp. 1–4.
- D. E. Arkadaş, M. Akpınar, E. G. Schmidt, and K. Werner Schmidt, “Clock Synchronization for the Controller Area Network using Bit Timing Information,” in Signal Processing and Communications Applications Conference, May 2022, pp. 1–4.