MESSAGE SCHEDULING FOR THE STATIC AND DYNAMIC SEGMENT OF FLEXRAY: ALGORITHMS AND APPLICATIONS

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

BY

ÖZGÜR KIZILAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2015

Approval of the thesis:

MESSAGE SCHEDULING FOR THE STATIC AND DYNAMIC SEGMENT OF FLEXRAY: ALGORITHMS AND APPLICATIONS

submitted by **ÖZGÜR KIZILAY** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Gönül Turhan Sayan
Head of Department, **Electrical and Electronics Engineering** _____

Assoc. Prof. Dr. Şenan Ece Güran Schmidt
Supervisor, **Electrical and Electronics Engineering, METU** _____

Assoc. Prof. Dr. Klaus Schmidt
Co-supervisor, **Mechatronics Engineering, Cankaya Univ.** _____

Examining Committee Members:

Assoc. Prof. Dr. Cüneyt F. Bazlamaçcı
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Şenan Ece Güran Schmidt
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. İlkay Ulusoy-Parnas
Electrical and Electronics Engineering, METU _____

Prof. Dr. Halit Oğuztüzün
Computer Engineering, METU _____

Prof. Dr. Semih BİLGEN
Computer Engineering, Yeditepe University _____

Date: **18.12.2015**

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, Last Name: ÖZGÜR KIZILAY

Signature :

ABSTRACT

MESSAGE SCHEDULING FOR THE STATIC AND DYNAMIC SEGMENT OF FLEXRAY: ALGORITHMS AND APPLICATIONS

Kızılay, Özgür
M.S., Department of Electrical and Electronics Engineering
Supervisor : Assoc. Prof. Dr. Şenan Ece Güran Schmidt
Co-Supervisor : Assoc. Prof. Dr. Klaus Schmidt

December 2015, 50 pages

Today's automobiles comprise an increasing number of electronic components. Some of these components are used for entertainment purposes and some of them are used for safety-critical application such as X-by-Wire that are implemented on electronic control units (ECUs). The safe data exchange among such ECUs has to be realized by a robust and reliable in-vehicle network protocol. In this context, FlexRay is one of the new generation in-vehicle network protocols which is already used in upper class series vehicles.

In principle, FlexRay enables the reliable transmission of periodic and sporadic messages that are generated by ECUs on two communication channels. Nonetheless, it is the user's responsibility to configure the FlexRay parameters. In particular, it is required to determine a message schedule such that each message meets its deadline.

In general, two types of messages – periodic and sporadic messages – are considered. In FlexRay, periodic messages are transmitted in the static segment and sporadic messages are transmitted in the dynamic segment. Since these two types of messages have

different timing characteristics and the corresponding segments in FlexRay have different arbitration properties, their scheduling has to be performed by different types of algorithms.

In this thesis, we focus on the message scheduling on the static segment and the dynamic segment of FlexRay. We first analyze practical requirements for the static segment scheduling based on previous studies that involve the solution of a linear integer problem (LIP). In order to circumvent the requirement for an LIP solver, a new heuristic static segment scheduling algorithm is developed and implemented. Its performance is evaluated by several test cases. Next, we consider the FlexRay dynamic segment. We first propose an improvement of an existing algorithm for the worst-case response time analysis and then develop a new algorithm for the priority assignment on the FlexRay dynamic segment. The practicability of the presented algorithms is established by various examples.

Keywords: FlexRay, Static segment, Dynamic segment, Message Scheduling

ÖZ

FLEXRAY VERİYOLU STATİK VE DİNAMİK BÖLÜTLERİ İÇİN MESAJ ÇİZELGELEMESİ: ALGORİTMALAR VE UYGULAMALAR

Kızılay, Özgür
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Assoc. Prof. Dr. Şenan Ece Güran Schmidt
Ortak Tez Yöneticisi : Assoc. Prof. Dr. Klaus Schmidt

Aralık 2015 , 50 sayfa

Günümüz otomobilleri her geçen gün daha fazla elektronik parça içermektedir. Bu elektronik parçaların bazıları eğlence amaçlı kullanılırken bazıları güvenlik ve X-by-Wire uygulamaları gibi daha kritik alanlarda kullanılmaktadır. Elektronik parçaların kullanımının olduğu yerde veri alışverişi kaçınılmaz hale gelmektedir. Elektronik kontrol ünitelerinin (EKÜ) güvenli bir şekilde haberleşebilmesi için sağlam ve güvenilir bir araç içi ağ protokolüne ihityaç vardır. FlexRay daha sık kullanılmaya başlanan yeni nesil araç içi ağ protokollerinden biridir.

FlexRay EKÜ'ler tarafından üretilen periyodik (periodic) ve sporadik (sporadic) mesajların güvenilir bir şekilde iki kanal üzerinden iletilmesini sağlamaktadır. FlexRay ağ parametre değerlerinin belirlenmesi ve her mesajın zaman sınırından önce gönderilmesini garantileyecek bir çizelge tasarımı yapılmalıdır.

Dönemleme açışından bakıldığında iki tip mesaj vardır, periyodik ve sporadik mesajlar. FlexRay'de periyodik mesajlar statik bölütte (static segment) ve sporadik me-

sajlar ise dinamik bölütte (dynamic segment) iletilmektedir. Bu iki tip mesaj farklı zamanlama karakteristiğine sahip olduğu için, bu mesajların çizelgelemeleri farklı tipte algoritmalar kullanılarak planlanmaktadır.

Bu tezde, biz bir FlexRay sisteminde statik bölüt ve dinamik bölüt mesaj çizelgelemesine odaklandık. Öncelikle statik bölüt çizelgelemesi için doğrusal tamsayı programlama (Linear Integer Programming) kullanan bir çözüm içeren önceki bir çalışmaya dayanarak pratik gerekleri analiz ettik. Doğrusal tamsayı programlama çözücü içermeyen bir keşifsel algoritma geliştirdik ve doğruladık. Algoritmanın performansını test durumlarıyla ölçtük. Daha sonra FlexRay dinamik bölüt üzerinde çalıştık. Öncelikle en kötü cevap zamanı analizi için daha önce ortaya konulmuş bir algoritmayı iyileştirdik ve daha sonra FlexRay dinamik bölüt üzerinde öncelik atayan yeni bir algoritma geliştirdik. Bu algoritmanın kullanışlılığını çeşitli örneklerle ortaya koyduk.

Anahtar Kelimeler: Flexray, Sabit bölüt, Hareketli bölüt, Mesaj Çizelgelemesi

To My Wife, Ayça

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisors Assoc. Prof. Dr. Şenan Ece Güran Schmidt and Assoc. Prof. Dr. Klaus Schmidt for supporting and guiding me with patience, enthusiasm and immense knowledge. I could not have imagined having better advisors and mentors for my M.S study.

Throughout my study, the biggest support came from my wife, Ayça. She helped me a lot both psychologically and physically. Without her support I would not complete this study. Also a special thanks to my dear brother and friends who always encouraged me, Özgün, Duygu, Hasan, Ceren, Gül, Volkan, Çağatay, Carol.

I would like to thank to my employer ASELSAN Inc. for the supporting my academic studies. My sincere thanks also goes to my colleague and team leader Dr.Mustafa DURSUN for his insightful comments, encouragement and full support. I am grateful to my colleagues Güner, Cihan, Nagehan, Serap, Hande and Ümit for their spiritual help which made me relaxed when I was stressfull.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF ABBREVIATIONS

ECU	Electronic Control Unit
CAN	Controller Area Network
LIN	Local Interconnect Network
FC	FlexRay Cycle
MT	Macro Tick
TDMA	Time Division Multiple Access
FID	Frame Identification
DS	Dynamic Segment
WCRT	Worst-Case Response Time
NIT	Network Idle Time
SW	Symbol Window
MS	Minislot

CHAPTER 1

INTRODUCTION

The automotive industry is one of the biggest industries in the world and to raise its share in the global market it needs to keep innovating. Today, the automotive industry presents automobiles with improved safety, comfort, performance and a reduced carbon foot-print. While improving automobiles, more and more electronics are getting involved and the number of electronic control units (ECUs) used in automobiles is getting larger. As the number of ECUs is increasing, the amount of data exchanged by these ECUs also increases. Additionally, data exchange reliability needs to be improved due to the fact that automobile's safety and control systems rely on the data coming from other ECUs. In the past, there were two very commonly used standards: controller area network (CAN) and local interconnect network (LIN). However, with the emerging needs of a higher bandwidths and data reliability for applications such as X-by-Wire, the FlexRay protocol was developed as an advanced embedded in-vehicle network by the FlexRay Consortium. Table 1.1 shows a comparison of the properties and usage areas of the three protocols: LIN, CAN and FlexRay.

Table1.1: Message Periods

Bus	LIN	CAN	FlexRay
Speed	$40kbit/s$	$1Mbit/s$	$10Mbit/s$
Cost	$	$$	$$$
Wires	1	2	2 *or* 4
Typical Applications	Body Electronics (Mirrors, Power Seats, Accessories)	Powertrain (Engine, Transmission, ABS)	High-Performance Powertrain, Safety (Drive-by-wire, active suspension, adaptive cruise control)

The basic operation of FlexRay is described in [2, 9, 8]. In comparison to existing protocols such as CAN [19], FlexRay offers two channels with a high bandwidth of 10Mbit/s. Its cyclic operation supports both time-triggered and event-triggered communication. In each FlexRay cycle, the static segment (SS) employs the idea of time division multiple access (TDMA) to enable the transmission of periodic messages in unique static slots, while sporadic messages can be sent in dynamic slots in the dynamic segment (DS). In this context, the proper utilization of FlexRay depends on the configuration of the message transmission by an appropriate message schedule.

The FlexRay SS requires to assign a frame ID (FID), a scheduling repetition and a scheduling offset to each periodic message so as to configure its transmitting static slots [16, 17]. The FIDs are limited and they are exclusively allocated to the nodes. Hence, for a given message set, decreasing the number of allocated FIDs per node is required for the extensibility of the network. An important fraction of the messages in automotive networks are generated periodically, however it is possible that the transmission of these messages deviates from this periodicity with some jitter because of the message schedule. The control applications generally rely on the periodic transmission of these messages. Therefore, it is beneficial to achieve message schedules with minimum jitter. Message scheduling on the DS of FlexRay requires assigning priority levels to the sporadic messages such that each message meets its specified deadline. That is, the worst-case response time (WCRT) between message generation and message reception has to be smaller than the deadline. Hereby, it has to be noted that the priority-based arbitration in the DS complicates the WCRT computation since the transmission of higher-priority messages can block, and hence delay, the transmission of lower-priority messages.

Referring to the stated requirements, this thesis contains three main contributions regarding the message scheduling on the FlexRay SS, the WCRT analysis on the FlexRay DS and the priority assignment on the FlexRay DS. These contributions are summarized as follows.

- The first contribution of this thesis concerns the assignment of FID, repetition and offset to each periodic message to be transmitted in the FlexRay SS. Existing work on this subject requires the solution of linear integer programs (LIPs).

This either requires the use of open source LIP solvers that cannot be integrated into proprietary software or the use of expensive professional LIP solvers. Accordingly, this thesis proposes a new heuristic to compute good quality assignments for the FlexRay SS without the need for an LIP solver. Case studies show the suitability of the proposed heuristic.

- The second contribution of this thesis concerns the WCRT analysis for sporadic messages transmitted in the FlexRay DS. It is based on the observation in [12] that the WCRT computation in [17] gives optimistic results in certain cases. In order to address this issue, the thesis proposes an improved WCRT analysis and demonstrates it by several test cases.

- The third contribution of this thesis concerns the priority assignment on the FlexRay DS. To this end, the algorithm in [18] is studied and it is observed that its run-time considerably increases for large message sets. Hence, this thesis proposes a new priority assignment algorithm with a lower run-time. The practicability of this algorithm is supported by various test cases.

The existing literature provides scheduling algorithms for the FlexRay SS and DS. [6] proposes a fast greedy heuristic and a Integer Linear Programming formulation to determine the optimal number of slots in the SS. [13] focuses on computing the optimal FlexRay cycle duration in order to minimize network delays. [7] proposes a modular framework with a symbolic representation that is used by an Integer Linear Programming (ILP) solver. The framework enables determining a schedule that respects all bus and processor constraints as well as end-to-end timing constraints. [21] defines an extensibility policy and extensibility evaluation for the FlexRay SS. Then, an optimal integer linear programming-based formulation and a fast heuristic algorithm are proposed as the basic message scheduling algorithms. [5] investigates the problem of allocating a minimum number of static slots to each FlexRay node, while guaranteeing that all periodic messages will be transmitted before their deadlines. It has to be noted that none of these approaches captures the trade-off between the FID allocation and the jitter as important performance metrics for in-vehicle communication. Although the previous work of Schmidt and Schmidt in [17] addresses this trade-off, the message schedule computation is carried out using a LIP formulation

with the stated disadvantages.

Regarding existing approaches for the FlexRay DS, the WCRT analysis is studied in [14, 18, 11, 20, 12]. In [14], it is considered that a fixed worst-case time interval for the analysis is given. In contrast, our algorithm iteratively increases the time interval under investigation by one FC until a deadline violation occurs (line 8) or schedulability can be verified (line 6). [11] presents an approach for the WCRT analysis of the FlexRay dynamic segment, which takes slot-multiplexing into account. [20] develops an approach for the WCRT analysis of the dynamic segment with slot multiplexing that returns less pessimistic results compared to previously known techniques. In addition, the work in [12] computes exact response times for the FlexRay DS with a very high computational complexity. The algorithm in [18] is the basis for the new WCRT algorithm proposed in this thesis. In particular, this thesis modifies the constraints in [18] in order to avoid an optimistic WCRT computation. Finally, the priority assignment for the FlexRay DS is only considered in [18, 22]. [22] studies the problem of timing analysis of frames transmitted in the FlexRay dynamic segment, providing a tight upper bound to the worst case response times. In addition they propose an algorithm to assign identifiers (priorities) to frames, to optimize a design objective. The proposed algorithm in [18] shows a considerable run-time for large message sets and an improvement is provided in this thesis.

The remainder of this thesis is organized as follows. In Chapter 2, the FlexRay protocol basics and its operation are specified. Chapter 3 is devoted to static segment scheduling. Firstly, scheduling requirements are described and formalized in the form of an objective function. The chapter then explains the optimal scheduling algorithm for the SS and the proposed heuristic algorithm. The WCRT analysis and priority assignment for the FlexRay DS is discussed in Chapter 4. Chapter 5 gives conclusions and outlines directions for future research.

CHAPTER 2

FLEXRAY NETWORK PROTOCOL

The Flexray protocol is developed by a consortium consisting of Adam Opel GmbH, Bayerische Motoren Werke AG, Daimler AG, Freescale Halbleiter Deutschland GmbH, NXP B.V., Robert Bosch GmbH, and Volkswagen AG [2]. The Flexray protocol is introduced due to requirements that are pointed out in [2, 8, 10]. As compared to its predecessor CAN [19], FlexRay offers a higher bandwidth of 10 Mbit/s with two channels and it is capable of dealing with both time-triggered and event-triggered messages. As depicted in the FlexRay Protocol specification [2], FlexRay operates in cycles. Scheduled messages are delivered on certain previously determined FlexRay cycles (FCs). The cycle count of FlexRay is identified by the value `vCycleCounter` which is a 6-bit value. Hence, FlexRay has up to $2^6 = 64$ FCs. This communication cycle scheme is depicted in Figure 2.1.

The communication cycle comprises the static segment, dynamic segment, symbol window and network idle time and cycles are executed repeatedly. The smallest time unit of the protocol is *macrotick* (MT) and it can take values between 1 μs and 6 μs. The duration of the FCs is predetermined and fixed: `gdCycle` = `gMacroPerCycle` $\cdot$ `gdMacrotick`, `gdMacrotick` is the value of MT and `gMacroPerCycle` is the number of MTs per FC. The time hierarchy may be seen in Figure 2.2.

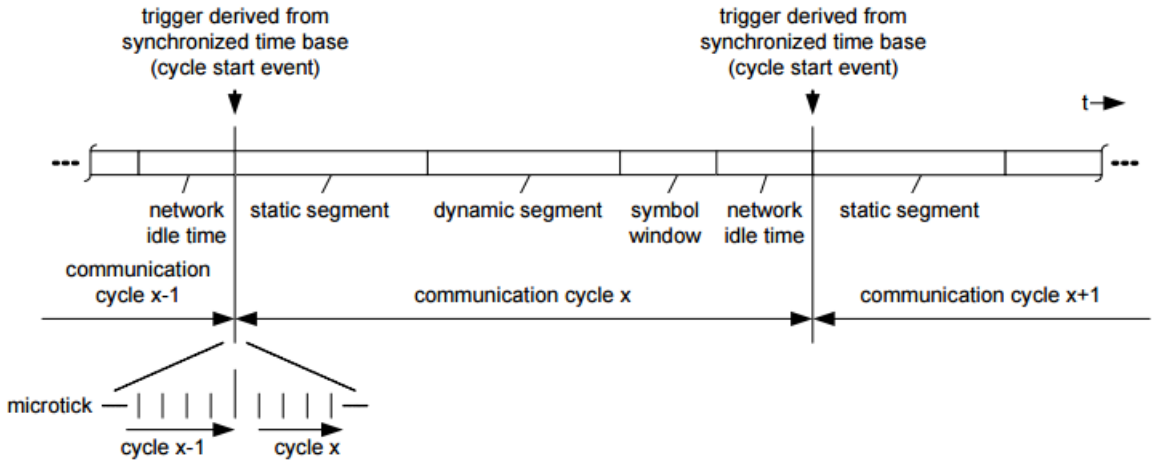

Figure 2.1: Time base triggered communication cycle [2].

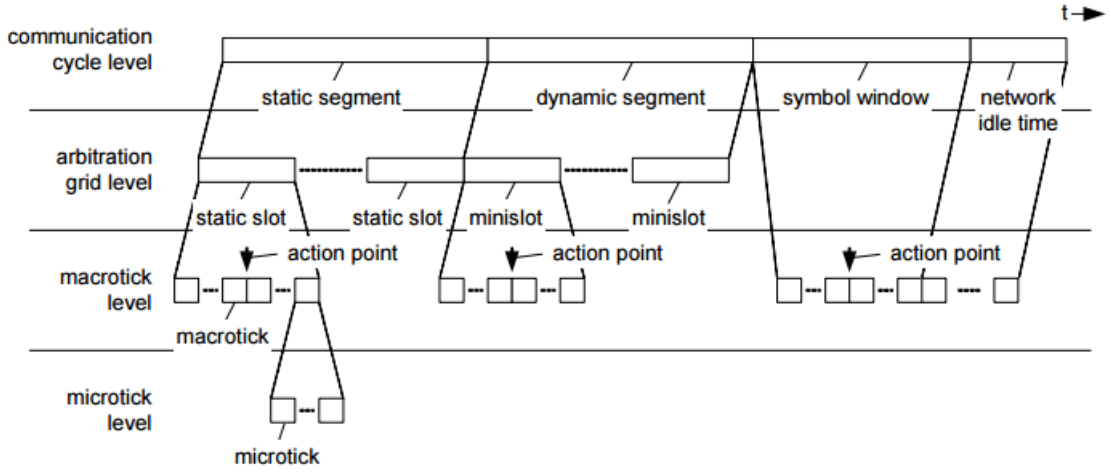

Figure 2.2: Timing hierarchy within the communication cycle[2]

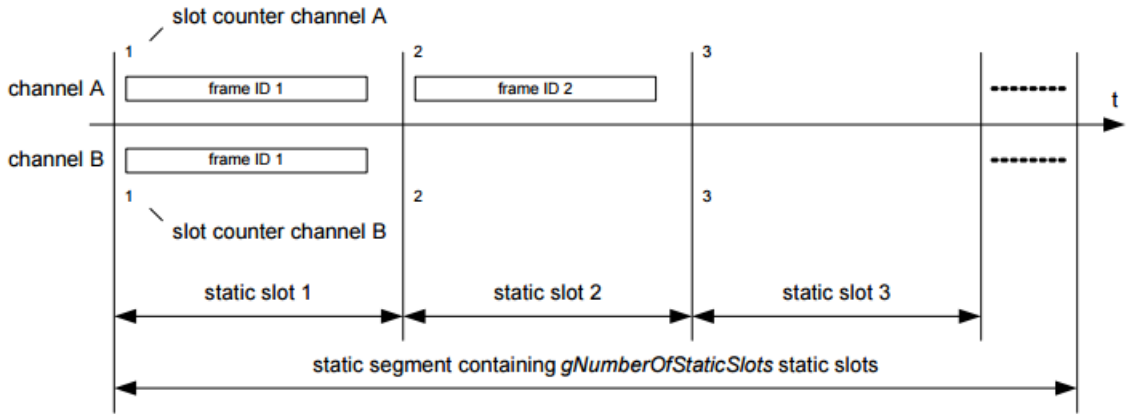

Figure 2.3: Structure of the static segment[2]

2.1 Static Segment

The static segment (SS) of FlexRay uses a TDMA scheme and is designed for the transmission of real-time periodic messages. The SS is divided into equal length time slots. The number of these time slots is `gNumberOfStaticSlots`. The duration of the time slots is `gdStaticSlot`. Therefore, the duration of the SS is evaluated as $\texttt{gNumberOfStaticSlots} \cdot \texttt{gdStaticSlot} \cdot \texttt{gdMacrotick}$. A state variable `vSlotCounter` is maintained by each node and by means of this variable, each node knows the executing slot number. The bus arbitration process in the SS makes use of the TDMA approach [4]. Each node has previously assigned FIDs, and when `vSlotCounter` is equal to the related node's FID, then that node makes its transmission. The FID is the slot number in the static segment to which a frame is assigned. In Figure 2.3, the SS timing characteristics are illustrated.

According to the protocol specification [2] each message in the SS has an unique triple (f_M, r_M, o_M) and each FID is assigned to a unique node. Assigning this triple to the messages is the scheduling process. f_M is the FID that message is assigned, r_M is the repetition value which denotes that the message is going to be transmitted every r_M FCs and o_M is the offset value which can be explained as the FC number in which the message is transmitted first.

In Figure 2.4, an example schedule with 3 messages is given. According to the pre-

FID / Cycle Count	1	2	3	....			
0	M1		M3	....	DS	SW	NIT
1	M1		M2	....	DS	SW	NIT
2	M1		M3	....	DS	SW	NIT
3	M1		M2	....	DS	SW	NIT
⋮				⋮			
63	M1		M2	....	DS	SW	NIT

Figure 2.4: Static Segment Scheduling Example

sented schedule, M1 is assigned to slot 1 with FID 1 with a repetition of 1. FID 3 is assigned to both M2 and M3, whereas their offset values are different; M3 has an offset 0 and M2 has an offset 1. More than 1 message can be assigned to the same FID with different offset values as in the case of this example. With the use of this kind of slot multiplexing, FIDs may be used more efficiently.

2.2 Dynamic Segment

In embedded systems, some of messages are real-time periodic messages and these messages should be transmitted in the SS in the Flexray operation in order not to miss their deadlines. On the other hand there are sporadic messages and these messages are transmitted in the dynamic segment (DS) of the Flexray operation. Sporadic messages are not transmitted in the SS because transmitting such occasional messages in the SS would result in an inefficient use of the Flexray cycle with allowing empty slots to pass frequently.

The DS of Flexray is divided into equal length *minislots* (MS). Each *minislot* is composed of `gdMinislot` number of macroticks. The number of *minislots* that each Flexray dynamic segment has is `gNumberOfMinislots`. The number of *minislots* may be between 0 and 7986 [2].

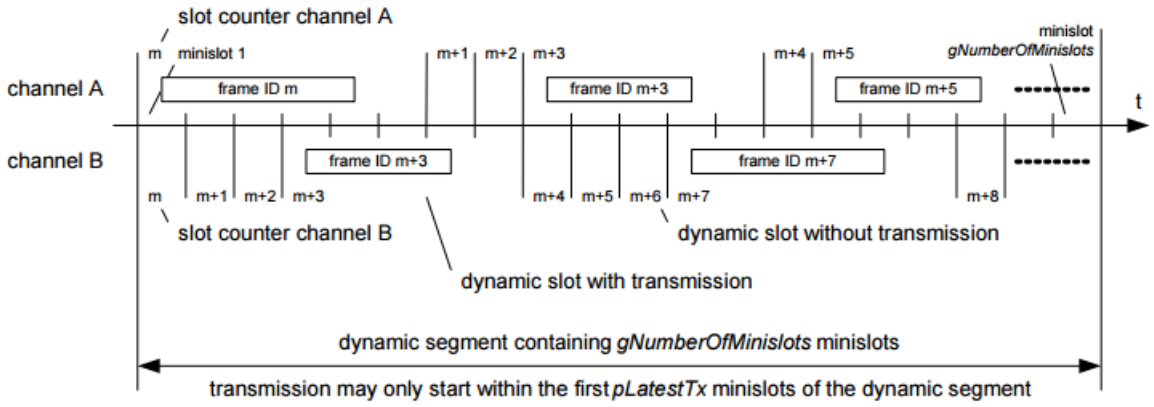

Figure 2.5: Structure of the Dynamic Segment [2]

The structure of the DS is depicted in Figure 2.5. In each channel, the slot counter accounts for the ID of the current slot. Slot IDs start from a pre-configured value. If there is a message to be transmitted with the ID of the current slot ID, then that message is transmitted and the duration of the slot is determined with the duration of the transmitted message. The durations of the messages are specified in multiples of MSs before the operation and scheduling. If there is not any message to be transmitted that has the ID of the current slot, then that slot remains empty with a duration of one MS. At this point, *minislot* should not be confused with dynamic slot; dynamic slot is an entity that may include a message transmission and in that case contains as many *minislots* as the message has and it may not include a message transmission and in that case it contains one *minislot*.

Each *minislot* has an action point that is some certain number of macroticks after the starting point of the *minislot*. The number of macroticks that characterizes the action point is `gdMinislotActionPointOffset`.

In the DS, the message transmissions start and end at the action points. This behavior is shown in Figure 2.6. A frame transmission is started at the action point of the first minislot and is ended at the corresponding last minislot of that dynamic slot.

The dynamic slot consists of two phases as may be seen at Figure 2.6. These phases are the transmission phase and the idle phase. The transmission phase is mandatory and it starts from the first minislot and it ends at the minislot in which the frame

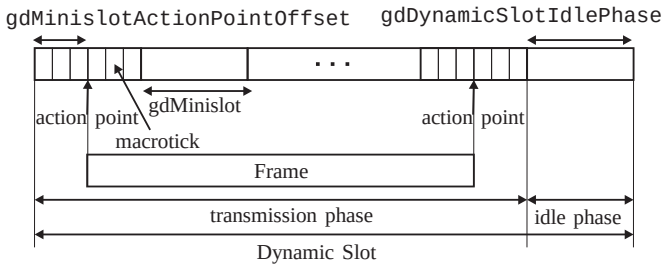

Figure 2.6: Structure of the Dynamic Slot [15]

transmission ends. The idle phase is optional. In the idle phase, no communication takes place and it provides time for all nodes to complete idle detection within the dynamic slot [2]. These both phases are composed of a certain number of MSs that are configured at the time of design of the Flexray network.

2.3 Symbol Window

In the Flexray cycle, after static and dynamic segments are completed, an optional symbol window part may be performed. In this part either a media acces test symbol (MTS) or a wake-up also during operation possible (WUDOP) may be sent [2].

The symbol window consists if a certain number of macroticks and that number `gdSymbolWindow` is configured.

As it may be seen in Figure 2.7, the transmission of the symbol starts at the action point of the symbol window which is an offset of `gdSymbolWindowActionPointOffset` macroticks long.

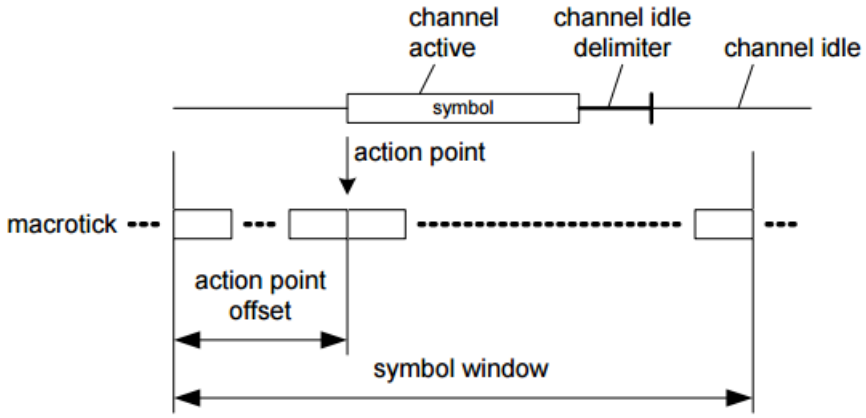

Figure 2.7: Timing within the symbol window [2]

2.4 Network Idle Time

In the Flexray cycle, the Network Idle Time (NIT) takes part after the SW. This part provides time for nodes to calculate and apply clock correction. This part is also used as a phase in which some communication related tasks are performed [2].

CHAPTER 3

FLEXRAY STATIC SEGMENT SCHEDULING

Periodic messages are transmitted in FlexRay static segment. The specific slots that messages are scheduled to be transmitted in are determined by the offline scheduling process. In this chapter, the scheduling methodology for the FlexRay static segment is discussed. Firstly, background of scheduling is presented followed by a discussion of the existing scheduling algorithm in [16] which is formulated and implemented making use of linear integer programming (LIP). Our contribution in this chapter is the design of a heuristic LIP-free scheduling algorithm with the following motivation. There are open source LIP tools and professional LIP tools in the market. On the one hand, an open source LIP tool is used, the resulting scheduling application has to be made open source also. On the other hand, professional LIP tools are expensive. Hence, we develop an LIP-free heuristic algorithm and show that it performs well in comparison to the LIP formulation in [16]. The chapter is concluded by a case study of the heuristic algorithm.

3.1 Background on Scheduling

The structure of the static segment of the FlexRay protocol is defined in Chapter 2. Scheduling messages that are transmitted in static segment basically requires assigning a unique triple of (f_M, r_M, o_M) to each message M. Here, f_M denotes the FID, r_M denotes the message repetition and o_m denotes the message offset. While assigning these free parameters to a set of messages, certain constraints apply and assumptions are made together with the constraints. After assumptions and constraints are for-

mulated, the solution of the scheduling problem depends on the chosen performance metrics. The performance metrics in [17] are allocating a minimum number of FIDs and minimizing the message jitter, that is, the deviation from the message periodicity.

3.1.1 Notation and Assumptions

The scheduling problem is formulated in the following setting: each node i has a set of periodic messages $\mathcal{M}^i$. The scheduling process assigns a triple (f_M, r_M, o_M) to each message $M \in \mathcal{M}^i$. Each periodic message M has a period p_M. The period p_M is assumed to be measured in multiples of the FC duration `gdCycle`.

Messages in the set $\mathcal{M}^i$ have deadlines which can be defined as the maximum time between the generation and reception of the message. As is commonly done for periodic messages, the deadline value is assumed to be equal to the period. The duration of a static slot `gdStaticSlot` is assumed to be chosen such that every message may be accommodated by one static slot.

3.1.2 Requirements

In this section, requirements that we need for a correct scheduling according to the FlexRay specification [2] are presented.

It is required that repetition value r_M should be in the range of $[1,64]$, whereby $r_M \in \{2^q | 0 \leq q \leq 6\}$. This requirement emerges from the cyclic operation of the FlexRay Protocol. Since FlexRay repeats its operation every 64 cycles, r_M may not exceed this value.

The offset value o_M should be strictly smaller than the repetition r_M, $0 \leq o_M < r_M$. Since o_M is defined as the cycle number in which the message is transmitted first, o_M can not exceed the repetition.

The total number of FIDs used should be smaller than the total number of slots in the static segment, $0 \leq f_M \leq$ `gNumberOfStaticSlots`. Since a static segment slot is assigned exactly one FID, the total number of FIDs also should be at most the number

of static segment slots.

In order for messages to meet their deadline, the repetition value should be less or equal than the period, $r_m \leq p_M$. Note that both r_m and p_M are multiples of the FC.

Each FID should be assigned to a unique node; an FID may not be assigned to messages from different nodes.

3.1.3 Performance Metrics

After assumptions are made and requirements that come from the FlexRay protocol specification [2] are pointed out, we discuss the performance metrics that characterize the scheduling algorithm. We use two performance metrics as introduced in [17]: minimizing the number of FIDs used and minimizing the jitter.

One of the performance metrics of our scheduling algorithm is to minimize the FIDs used. This metric is offered in order to keep the used portion of the static segment as small as possible. Using a minimum portion from the static segment would allow users to add extra messages with ease if necessary. When a message $\mathcal{M}^i$ of from node i is assigned to a slot with a repetition of r_M, the fraction of the usage of that slot may be evaluated as in equation (3.1).

$$A_M = \frac{1}{r_M}. \tag{3.1}$$

The repetition value r_M specifies that message M will be transmitted every r_M cycles in its assigned slot. That slot is hence going to be used by message M every r_M cycles out of 64 cycles. For example if $r_M = 2$, the message is going to use the slot every 2 cycles, the utilization of that slot by message M is evaluated as $A_M = \frac{1}{2}$ and it holds that 32 out of 64 cycles are used by M.

Following the proposition in [17], the expression for the FID allocation of node i becomes

$$FA^i := \lceil \sum_{M \in \mathcal{M}^i} A_M \rceil = \lceil \sum_{M \in \mathcal{M}^i} \frac{1}{r_M} \rceil. \tag{3.2}$$

The ceiling function in (3.2) exists to ensure that the FID allocation is an integer. Since FID allocation determines the total number of FIDs, it must be an integer.

Minimizing the jitter is the other performance metric of our scheduling algorithm. Jitter is defined as the message's deviation from periodicity. Quantification of the jitter is made in [17] by defining a relative jitter J_M for a message $M \in \mathcal{M}^i$:

$$J_M := \frac{2 \cdot (r_M - b) \cdot b}{p_M \cdot r_M}, \tag{3.3}$$

where b is defined as

$$b = p_M \bmod r_M, \;\; 0 \leq b < r_M. \tag{3.4}$$

The jitter defined in (3.3) is a relative quantity which allows us to use the term as an indication of jitter and compare the jitters created by different r_M values.

The specified performance metrics for the scheduling algorithm lead us to the objective function which contains these two performance metrics. If we have a set of messages $\mathcal{M}^i$ from a node i, then to schedule these messages such that they use the minimum number of FIDs and they experience minimum jitter, we should minimize FA^i in (3.2) and the relative jitter sum

$$J^i := \sum_{M \in \mathcal{M}^i} J_M. \tag{3.5}$$

Note that both terms depend on the choice of r_M for each message. [16] formulates an LIP to decide on the optimum values of repetitions for the joint minimization of FID utilization and jitter. A set of potential repetitions for messages is introduced: $\mathcal{R}_M = \{2^q | 2^q \leq p_M\}$. For the LIP formulation, a boolean decision variable $x_{M,r}$ for each $r \in \mathcal{R}_M$ is introduced. This boolean variable gets the value 1 if $r_M = r$ and 0 otherwise. The sum of these boolean variables for one message should be 1, since a message is assigned a unique repetition value. (3.6) specifies this constraint.

$$\sum_{r \in \mathcal{R}_M} x_{M,r} = 1. \tag{3.6}$$

Using (3.3), the total jitter J_M is evaluated as

$$J_M = \sum_{r \in \mathcal{R}_M} x_{M,r} \cdot \frac{2 \cdot (r - (p_M \bmod r)) \cdot (p_M \bmod r))}{p_M \cdot r}. \tag{3.7}$$

Using (3.1), the FID allocation for node i is

$$FA^i = \lceil \sum_{M \in \mathcal{M}^i} \sum_{r \in \mathcal{R}_M} \frac{x_{M,r}}{r} \rceil = \lceil (64 \cdot \sum_{M \in \mathcal{M}^i} \sum_{r \in \mathcal{R}_M} \frac{x_{M,r}}{r}) / 64 \rceil.$$

Here, the expression contains a ceiling function which cannot be expressed in an LIP. Therefore, the expression may be written as in (3.8) to account for the ceiling function.

$$FA^i \cdot 64 = 64 \cdot (\sum_{M \in \mathcal{M}^i} \sum_{r \in \mathcal{R}_M} \frac{x_{M,r}}{r}) + c. \tag{3.8}$$

In (3.8), the dummy integer variable c is introduced. When c is minimum, FA^i evaluates to the ceiling value.

The optimization function which jointly minimizes the FID allocation and jitter is presented in (3.9). In this joint objective function , γ_{FA} is the weight of the FID allocation part and γ_J is the weight of the jitter part. By changing these weight parameters, the user may decide which part should contribute more to the objective function.

$$\min_X \gamma_{FA} FA^i + \gamma_J \cdot \sum_{M \in \mathcal{M}^i} J_M. \tag{3.9}$$

The assignment of r_M is determined using the objective function (3.9). Using this repetition assignment, the assignment of FIDs and offsets can be performed making direct use of Algorithm 1.

input : L_i, f_{init}; Variable: $O = \{0, \ldots, 63\}$, $f_c := f_{\text{init}}$, $util := 0$
while L_i is not empty 1
  if $util = 1$ 2
    $util := 0$; $O := \{0, \ldots, 63\}$, $f_c := f_c + 1$ 3
  remove the first element M from L_i 4
  $util := util + 1/r_M$ 5
  assign $f_M := f_c$ 6
  assign smallest element o in O to o_M: $o_M := o$ 7
  remove all elements $o + k \cdot r_M$ from O for $k \in \mathbb{N}_0$ 8
return triple (r_M, o_M, f_M) for each $M \in \mathcal{M}^i$ 9

Algorithm 1: FID and offset assignment algorithm in [17].

3.2 Heuristic Scheduling Algorithm

In this part, the heuristic LIP-free static segment scheduling algorithm which is the main contribution of this thesis regarding the FlexRay static segment is presented.This heuristic algorithm is designed to get an FID assignment that is as close as possible to the optimal assignment in defined in (3.9). In a vehicle network a lot of nodes need to connect to bus. In order to schedule whole network, all periodic messages from all nodes need to be assigned an FID. While defining the objective function, we reduced our focus on a single node due to the requirement that "Each FID should be assigned to a unique node". This requirement enables us to apply our objective function to all nodes separately because free parameters of messages from different nodes do not have an impact on the other node's objective function. When the schedule of each node is computed, we obtain a solution of whole system.

Consider a node i and its set of messages $\mathcal{M}^i$. We define $r_{M_{max}}$ as the maximum possible repetition value that message $M \in \mathcal{M}^i$ can take without violating the deadline and $r_{M_{min}}$ as the repetition value that gives minimum relative jitter. Since the deadline is equal to the period and the period is in multiples of cycle time, $r_{M_{max}}$ is evaluated as:

$$r_{M_{max}} = 2^{i^{max}}, \text{ where } i^{max} \text{ is the maximum value satisfying } 0 \leq i^{max} \leq 6, \quad 2^{i^{max}} \leq p_M \tag{3.10}$$

$r_{M_{min}}$ is determined making use of the (3.3) as

$$r_{M_{min}} = r_i, \textit{ where } 1 \leq r_i \leq 64 \text{ is the value minimizing } J_{M_i} = \frac{2 \cdot (r_i - p_M \bmod r_i) \cdot p_M \bmod r_i}{p_M \cdot r_i}. \tag{3.11}$$

We next present our heuristic scheduling algorithm that we refer as Algorithm (4) to find the set of repetitions r_M which result in good (small) objective function value. We then employ Algorithm 1 in [17] to completely specify the triple of (f_M, r_M, o_M) for the scheduling. Algorithm (4) makes use of Algorithms 2 and 3.

To this end, we first determine $r_{M_{min}}$ and $r_{M_{max}}$ values and then specify an initial repetition value according to procedure in Algorithm 2 (lines 4-16). Algorithm 2 makes use of the functions 3.12 and 3.13. The resulting value for each message is the

repetition value that minimizes 3.13.

$$ComputeRelativeJitter(\gamma_J, r_M, p_M) = \gamma_J \times \frac{2 \cdot (r_M - p_M \bmod r_M) \cdot p_M \bmod r_M}{p_M \cdot r_M} \tag{3.12}$$

$$ComputeObjScore(\gamma_{FA}, \gamma_J, r_M, p_M) = \gamma_{FA} \times \frac{1}{r_M} + ComputeRelativeJitter(\gamma_J, r_M, p_M) \tag{3.13}$$

input : Message Set $\mathcal{M}^i$
output: Repetition Set

for *each* $M \in \mathcal{M}^i$ **do**
  compute $r_{M_{min}}$ and $r_{M_{max}}$
end
for *each* $M \in \mathcal{M}^i$ **do**
  $r_{opt} = r_{M_{max}}$
  $optScore = ComputeObjScore(\gamma_{FA}, \gamma_J, r_{opt}, p_M)$
  while $r_{opt} != r_{M_{min}}$ **do**
    $tempScore = ComputeObjScore(\gamma_{FA}, \gamma_J, \frac{r_{opt}}{2}, p_M)$
    if $tempScore < optScore$ **then**
      $r_{opt} = \frac{r_{opt}}{2}$
      $optScore = tempScore$
    else
      break
    end
  end
end

Algorithm 2: Starting Repetition Computation

After running Algorithm 2, a repetition value r_M is obtained for each message $M \in \mathcal{M}^i$ from which the o_M and f_M values may be computed. However, at this step, the repetition values do not give the minimum objective function value. The first term of the objective function (3.9) $FA^i = \lceil \sum_{M \in \mathcal{M}^i} \frac{1}{r_M} \rceil$ employs a ceiling function. If the term inside the ceiling function for the computed repetitions is not an integer, then decreasing

some of the repetition values would not change the value of FA^i. For example, if this term is 4.5, then some of the repetitions may be made smaller until the value reaches 5 without altering the term's contribution to the overall objective function result. By employing this fact, some certain repetitions which do not assume the value $r_{M_{min}}$ may be made smaller in order to reduce the jitter term of the objective function (3.9). To apply this idea, Algorithm 4 is proposed. Firstly, if there is margin in FA^i, the repetition that has the biggest impact on jitter term of the objective function is chosen. Then the chosen repetition is set as the new repetition of the message. The algorithm iterates until the term inside the ceiling of FA^i assumes an integer value or no more repetitions can be decreased. It employs 3 to find the best-cost message during these iterations.

input : Message Set $\mathcal{M}^i$, $margin$, γ_J, γ_{FA}
output: Best-cost message

for *each* $M \in \mathcal{M}^i$ **do**
  $r_{temp} = \frac{r_i}{2}$
  while $r_{temp} \geq r_{M_{min}}$ **do**
    $absCost_i = \frac{1}{r_{temp}} - \frac{1}{r_i}$
    $cost_i = (\frac{1}{r_{temp}} - \frac{1}{r_i}) \cdot \gamma_{FA}$
    $benefit_i = (ComputeRelativeJitter(\gamma_J, r_i, p_M) - ComputeRelativeJitter(\gamma_J, r_{temp}, p_M)) \cdot \gamma_J$
    if $\frac{cost_i}{benefit_i} < bestcost$ & $margin > absCost_i$ **then**
      $bestcost = \frac{cost_i}{benefit_i}$
      $best-cost\ message = M$
    end
    $r_{temp} = \frac{r_i}{2}$
  end
end

Algorithm 3: Finding Best-Cost Message

After the application of Algorithm (4), the set of repetitions which result in a better (smaller) objective function value is determined. In order to complete scheduling, the triple of (f_M, r_M, o_M) should be specified. Since the r_M parameters are already determined, Algorithm 1 in [17] is used.

input : Message Set $\mathcal{M}^i$, γ_J, γ_{FA}

output: Repetition Set

```
1  while true do
2      margin = ⌈∑_{M∈M^i} 1/r_M⌉ − ∑_{M∈M^i} 1/r_M
3      if margin > 0 then
4          Find best-cost message with Algorithm 3
5          if found then
6              change the repetition of the message with the one found in line 4
7          else
8              break
9          end
10     else
11         break
12     end
13 end
```

Algorithm 4: Perform Optimization

Table3.1:Message Periods

	Period in ms
m_1	10
m_2	70
m_3	150
m_4	770
m_5	340

We next evaluate our heuristic algorithm using an example for illustration. The scheduling computation for the set of messages presented in Table 3.1 is executed. The FlexRay configuration in this case is *cycle time* : 10 ms, $\gamma_J = 1$, $\gamma_{FA} = 1$. Firstly, Algorithm 2 is executed. The output of this algorithm is the initial repetition set which is shown in Table 3.2. These repetition values determine the FID part of the objective function: $\frac{1}{1} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} = 1.46875$. Since this value is not an integer, Algorithm 4 is applied in order to reduce the objective function value.

As the result, we obtain the repetition set in Table 3.3. The FID part of the objective function: $\frac{1}{1} + \frac{1}{2} + \frac{1}{8} + \frac{1}{8} + \frac{1}{4} = 2$. FID and offset assignments are computed using

Table3.2:Initial Repetiton Set

	Repetition
m_1	1
m_2	4
m_3	8
m_4	16
m_5	32

Algorithm 1 and results are given in Table 3.3.

Table3.3:Resulting Schedule

	Repetition	FID	Offset
m_1	1	1	0
m_2	2	2	0
m_3	8	2	3
m_4	8	2	7
m_5	4	2	1

Note that scheduling algorithm of [16] is optimal but our heuristic algorithm might not be optimal. Nevertheless, for the case presented, the resulting assignment of triple (f_M, r_M, o_M) and objective function value are identical to the values of the optimal algorithm.

3.3 Case Study

In this section, a case study of static segment message scheduling is presented. In this example message set is composed of 35 messages whose periods/deadlines are given in Table 3.4. Let `gdCycle` be 10 ms. We compute a schedule with $\gamma_J = 1$, $\gamma_{FA} = 1$ and $\gamma_J = 1$, $\gamma_{FA} = 2$.

The result found by running the heuristic algorithm with an input of $\gamma_J = 1$, $\gamma_{FA} = 1$ is given in the Table 3.5. According to results, in total 5 FIDs are used, an objective function value of 7.583071 is obtained. When this case is executed with optimal algorithm, the same objective function value is obtained thus in this case heuristic algorithm results in an optimal solution. Table 3.6 shows results computed by the heuristic algorithm with an input of $\gamma_J = 1$, $\gamma_{FA} = 2$. In that case 6 FIDs are used and objective function value is 10.019142. The optimal algorithm gets an objective

Table3.4:Message Periods

Message	Period in ms	Message	Period in ms	Message	Period in ms
m_1	10	m_2	70	m_3	150
m_4	770	m_5	340	m_6	500
m_7	40	m_8	90	m_9	240
m_{10}	840	m_{11}	630	m_{12}	570
m_{13}	700	m_{14}	400	m_{15}	150
m_{16}	210	m_{17}	310	m_{18}	320
m_{19}	440	m_{20}	520	m_{21}	270
m_{22}	280	m_{23}	290	m_{24}	110
m_{25}	120	m_{26}	130	m_{27}	140
m_{28}	470	m_{29}	480	m_{30}	490
m_{31}	610	m_{32}	620	m_{33}	680
m_{34}	750	m_{35}	760		

function value of 9.971413, the result of heuristic algorithm is not optimal.

We further performed tests with randomly generated message sets in order to compare the performance of the heuristic algorithm and the optimal LIP solution. For each message count at Table 3.7, we randomly generated 1000 test cases and evaluated the results. Randomly generated message periods are in between $30 - 10000\ ms$ and cycle is set to $10\ ms$. In the second column of Table 3.7, we present the number of cases in which two algorithms have the same result out of 1000 cases. In the third and fourth columns, we present average and maximum percentage differences between the objective function values of the two algorithms in 1000 test cases respectively.

The results in Table 3.7 show that average difference from the optimal algorithm is below %10 thus our heuristic algorithm may be considered as a good approximation to optimal algorithm. For some message counts, the maximum difference can be large; it may be said that there is a chance that computation can result in a difference of %47. If it is critical for the user not to have such a difference from the optimal result; exhaustive enumeration may be used in order to avoid usage of LIP.

Table3.5:Resulting Schedule, $\gamma_J = 1$, $\gamma_{FA} = 1$

Message	Repetition	FID	Offset	Message	Repetition	FID	Offset
m_1	1	1	0	m_2	4	2	0
m_3	8	4	0	m_4	16	5	1
m_5	32	5	13	m_6	16	5	2
m_7	4	2	1	m_8	8	4	1
m_9	8	4	2	m_{10}	16	5	3
m_{11}	32	5	14	m_{12}	8	4	3
m_{13}	8	4	4	m_{14}	8	4	5
m_{15}	8	4	6	m_{16}	4	2	2
m_{17}	16	5	4	m_{18}	32	5	15
m_{19}	4	2	3	m_{20}	16	5	5
m_{21}	8	4	7	m_{22}	4	3	0
m_{23}	16	5	6	m_{24}	4	3	1
m_{25}	4	3	2	m_{26}	4	3	3
m_{27}	8	5	0	m_{28}	16	5	7
m_{29}	16	5	9	m_{30}	16	5	10
m_{31}	32	5	29	m_{32}	32	5	30
m_{33}	32	5	31	m_{34}	16	5	11
m_{35}	16	5	12				

Table3.6:Resulting Schedule, $\gamma_J = 1$, $\gamma_{FA} = 2$

Message	Repetition	FID	Offset	Message	Repetition	FID	Offset
m_1	1	1	0	m_2	4	2	1
m_3	8	5	1	m_4	16	6	2
m_5	32	6	13	m_6	16	6	3
m_7	4	2	3	m_8	8	5	2
m_9	8	5	3	m_{10}	16	6	4
m_{11}	32	6	14	m_{12}	8	5	5
m_{13}	8	5	6	m_{14}	8	5	7
m_{15}	8	6	0	m_{16}	4	3	0
m_{17}	16	6	5	m_{18}	32	6	15
m_{19}	4	3	1	m_{20}	4	3	2
m_{21}	4	3	3	m_{22}	4	4	0
m_{23}	4	4	1	m_{24}	4	4	2
m_{25}	4	4	3	m_{26}	4	5	0
m_{27}	2	2	0	m_{28}	16	6	6
m_{29}	16	6	7	m_{30}	16	6	10
m_{31}	32	6	29	m_{32}	32	6	30
m_{33}	8	6	1	m_{34}	16	6	11
m_{35}	16	6	12				

Table3.7: Comparison of optimal LIP result and heuristic algorithm for different message counts.

Message Count	Exact Match # of Cases	Avarage Difference (%)	Max. Difference (%)
5	817	0.03	1.05
10	659	0.08	2.49
20	603	1.00	47.10
30	395	8.23	42.81
40	198	6.45	28.00
50	465	1.02	17.85
60	337	2.35	15.12
70	223	2.60	12.64
80	322	0.92	10.15
90	281	1.18	7.82
100	184	1.30	7.59

CHAPTER 4

DYNAMIC SEGMENT ANALYSIS AND SCHEDULING

In a FlexRay cycle, periodic and sporadic messages are transmitted according to the schedule and configuration parameters of the FlexRay Protocol. Scheduling messages and configuration of the FlexRay Protocol parameters are off-line processes that are executed before the actual implementation of FlexRay Network. After discussing scheduling for the FlexRay static segment in Chapter 3, this chapter focuses on scheduling of dynamic segment (DS) messages.

The FlexRay DS is intended for the transmission of sporadic messages. According to the description in Section 2.2, scheduling of DS messages addresses the problem of assigning an FID to each sporadic message such that no message misses its deadline. The procedure of assigning FIDs also depends on the configuration parameters of the FlexRay protocol. The dynamic segment is designed such that its length is usually not sufficient for accommodating all the sporadic messages in order to efficiently use the available bandwidth: since sporadic messages are not released periodically, reserving a dynamic slot with the length of each message in each cycle is not efficient. Nevertheless, each sporadic messages still has to meet its deadline.

In the DS, the FIDs assigned to sporadic messages can be considered as priorities. At the beginning of the DS, if the message with the highest priority (smallest assigned FID) is present, then that message is transmitted and the dynamic slot uses the number of FIDs corresponding to the message length. If it is not present, then the dynamic slot lasts exactly one MS. Afterwards the second highest priority message is checked and if present it is transmitted. This procedure goes on until the end of the DS. A message with a low priority may not be transmitted due to present high-priority messages. In

order to be certain that no message misses its deadline, the worst-case scenario for every message should be computed and it must be shown that, under the worst-case condition, every message meets its deadline.

This chapter focuses on two main topics. First, the computation of the worst-case response time (WCRT), that is, the longest time between message generation and reception for the FlexRay DS is considered. Second, the problem of assigning FIDs to sporadic messages such that the WCRT time is smaller than the message deadline is studied. Here, the main contributions are the improvement of the WCRT computation in [18] and a new FID assignment algorithm with a reduced run-time compared to existing algorithms.

This chapter is organized as follows. The notation is introduced and the assumptions made for the formulation of the problem are stated in Section 4.1. Then, the WCRT analysis is discussed in Section 4.2. In this context, first, the existing analysis is explained and then the improved analysis is put forward. and executed dynamic segment scheduling cases are going to be presented. Chapter ends with the discussion of dynamic segment scheduling.

4.1 Notation and Assumptions

In this section, the notation that is used to formalize the analysis is given and the assumptions made are discussed.

In the analysis an easier to follow notation compared to protocol specification notation in Section 2.2 is used. FlexRay operates in cycles. The duration of one cycle is `gdCycle` [2]. We call this parameter T_c throughout the analysis. The FlexRay dynamic segment is divided into equal length minislots. The duration of each minislot is `gdMinislot` macroticks, denoted as T_{MS}. The number of minislots in a dynamic segment is fixed and it is `gNumberOfMinislots` written as N_{MS}. The durations of Symbol Window and Network Idle Time are `gdSymbolWindow` and `gdNIT`. These parameters are denoted as T_{SW} and T_{NIT}, respectively.

In the DS, a message may be transmitted if the latest transmission point is not reached,

yet [2]. The latest transmission point is the beginning of the minislot such that the longest message does not fit in the remaining minislots. The corresponding parameter pLatestTx is calculated as

$$\texttt{pLatestTx} = \texttt{gNumberOfMinislots} - \texttt{aMinislotPerDynamicFrame} + 1,$$

where aMinislotPerDynamicFrame is the number of minislots needed for the transmission of the longest message [18].

In the DS, scheduling includes assigning FIDs to all messages and also determining the number of minislots. Then, the WCRT analysis is based on the already assigned FIDs and number of minislots. That is, we assume for the analysis that FIDs are already assigned to each message. We use the message set $\mathcal{D}$. In analogy to the studies in [3, 14, 18] we characterize each message $D \in \mathcal{D}$ by its *deadline* d_D, its *minimum inter-arrival time* p_D and its *payload* length b_D in two-byte words. Hereby, p_D models the minimum time between two consecutive message generations of the sporadic message D. In addition, we use the mapping between messages and their FIDs [18] as:

$$a_{\mathrm{DS}} : \mathcal{D} \rightarrow \{1, \ldots, \texttt{gNumberOfMinislots}\} \tag{4.1}$$

that determines the MS ID $a_{\mathrm{DS}}(D)$ for each message $D \in \mathcal{D}$.

According to the FlexRay specification [2] the frame length is evaluated as:

$$\texttt{FrameLength[gdBit]} = \texttt{PayloadLength} \cdot 20 + 94. \tag{4.2}$$

In equation 4.2, gdBit is the bit time in the channel. In the DS, frames are sent within the minislots and each message occcupies a certain number of minislots. This number is evaluated as:

$$\texttt{MinislotPerDynamicFrame[MS]} = 1 + \lceil \tfrac{0.1003 \cdot \texttt{gdBit} \cdot (\texttt{FrameLength}+1)}{\texttt{gdMacrotick} \cdot \texttt{gdMinislot}} \rceil + \texttt{gdDynamicSlotIdlePhase}. \tag{4.3}$$

In (4.3), gdDynamicSlotIdlePhase is the duration of the idle phase in each minislot

as described in Chapter 2.

While performing the WCRT analysis, we assume that each FID is assigned to a unique message such that FIDs are not shared by different messages. We make the analysis for a single channel of FlexRay, accounting for the fact that the FID assignment for the messages assigned to the two different FlexRay channels is independent.

4.2 Worst-Case Response Time Analysis

In this section we perform the analysis of the worst-case response time (WCRT) that a message scheduled in the FlexRay DS may experience. The WCRT depends on the FID assignment and the number of minislots in the DS. Firstly, the existing work in [18] is presented and discussed. Afterwards, improvements on the existing work are studied and finally case studies are presented.

4.2.1 Existing Analysis and Discussion

The WCRT for DS messages may be analyzed with two components. Consider a message $D \in \mathcal{D}$. The first component of the delay is experienced in the first FC of its generation. Since we are investigating the WCRT, we assume that the message D came just after the its assigned FID $a_{\mathrm{DS}}(D)$. In that case, the initial delay that D experiences in the first FC is:

$$t_{D,\mathrm{init}} = (N_{\mathrm{MS}} - a_{\mathrm{DS}}(D) + 1)T_{\mathrm{MS}} + T_{\mathrm{SW}} + T_{\mathrm{NIT}}. \tag{4.4}$$

The second component of the delay occurs due to blocking by higher-priority messages. If the DS of a FlexRay cycle can be filled with higher-priority messages up to `pLatestTx`, N_{latest}, then the message D cannot be transmitted in that cycle. The WCRT is experienced if the maximum number of cycles are filled with higher-priority messages. In the analysis of [18], an LIP formulation is presented. The boolean decision variables $x_{D',1}, \ldots, x_{D',f}$ for each message $D' \in \mathcal{D}$ are introduced. The messages $D' \in \mathcal{D}$ are higher-priority messages than D such that $a_{\mathrm{DS}}(D') < a_{\mathrm{DS}}(D)$. If the

boolean variable $x_{D',i} = 1$, then it means that the message D' is transmitted in FC i and if $x_{D',i} = 0$ then it means that the message D' is not transmitted in FC i, $1 \leq i \leq f$. As previously stated, p'_D is the minimum inter-arrival time between two instances of message D'. In [18], it is assumed that the maximum number of instances D' may be transmitted in j consecutive cycles is

$$\lceil j \cdot T_c / p_{D'} \rceil. \tag{4.5}$$

In this respect, for each cycle j with $1 \leq j \leq f$, $f - j + 1$ constraints for each message $D' \in \mathcal{D}$ are introduced. (4.6) states these constraints for each message.

$$\begin{aligned} x_{D',1} + \cdots + x_{D',j} &\leq \lceil j \cdot T_c / p_{D'} \rceil \\ \vdots \quad & \vdots \quad \vdots \\ x_{D',f-j+1} + \cdots + x_{D',f} &\leq \lceil j \cdot T_c / p_{D'} \rceil. \end{aligned} \tag{4.6}$$

Using the boolean variables $x_{D',j}$ for the messages $D' \in \mathcal{D}$ and cycles $1 \leq j \leq f$, the constraint such that message D is not transmitted in the cycles before than its scheduled cycle f is

$$\begin{aligned} &(N_{\text{latest}} - 1) \cdot T_{\text{MS}} < \\ &\sum_{\substack{D' \\ a_{\text{DS}}(D') < a_{\text{DS}}(D)}} x_{D',i} \cdot N_{D',\text{MS}} \cdot T_{\text{MS}} + (1 - x_{D',i}) \cdot T_{\text{MS}} \leq T_{\text{DS}}. \end{aligned} \tag{4.7}$$

For message D not to be transmitted in any cycle before f, each of the cycles $1 \leq i < f$ should be filled at least up to minislot N_{latest}. The term $x_{D',i} \cdot N_{D',\text{MS}}$ stands for the transmitted message D' in cycle i and $(1 - x_{D',i}) \cdot T_{\text{MS}}$ stands for the messages that are not transmitted in cycle i such that a minislot is passed empty. The upper limit in (4.7) is the overall duration of the DS.

In the final cycle f, the sum of the durations of the messages transmitted should also be less than T_{DS}:

$$\sum_{\substack{D' \\ a_{\text{DS}}(D') < a_{\text{DS}}(D)}} x_{D',f} \cdot N_{D',\text{MS}} \cdot T_{\text{MS}} + (1 - x_{D',f}) \cdot T_{\text{MS}} \leq T_{\text{DS}}. \tag{4.8}$$

The objective is then to maximize the cycle number in which message D cannot be transmitted. In this LIP formulation the objective corresponds to the maximization of the higher-priority message transmission time in the last cycle f. It can be expressed as:

$$J_{D,f} = \sum_{\substack{D' \\ a_{\mathrm{DS}}(D') < a_{\mathrm{DS}}(D)}} x_{D',f} \cdot N_{D',\mathrm{MS}} \cdot T_{\mathrm{MS}} + (1 - x_{D',f}) \cdot T_{\mathrm{MS}}. \tag{4.9}$$

If function $J_{D,f}$ is maximized, then the WCRT for massage D may be obtained. Then optimization problem in [18] is hence expressed as follows:

$$J^*_{D,f} = \max_X J_{D,f} \tag{4.10}$$

(4.10) is subject to the constraints (4.6), (4.7) and (4.8). If the message D may be transmitted in cycle f, then there should be room for the message D in the cycle: $J^*_{D,f} \leq (N_{\mathrm{latest}} - 1) \cdot T_{\mathrm{MS}}$ should be satisfied. The total delay that message D experiences is:

$$w_D = t_{D,\mathrm{init}} + (f-1) \cdot T_{\mathrm{c}} + T_{\mathrm{SS}} + J^*_{D,f} + N_{D,\mathrm{MS}} \cdot T_{\mathrm{MS}}. \tag{4.11}$$

This formulation is applied for all the sporadic messages. It starts iterating from cycle 1 and it ends at cycle $f = \lceil d_D / T_{\mathrm{c}} \rceil$. If it ends before reaching f, it means that message D may be scheduled without missing its deadline. But if the last iteration is reached, it means that message D misses its deadline.

The analysis presented here [18] depends on the consecutive occurrence limits of messages in (4.5). Nevertheless, as indicated in [12], this constraint is on inter-arrival times and it is also used as a constraint for inter-departure times. Hence, it implicitly assumes that for a given priority, the higher priority messages cannot be transmitted on the bus more frequently than their minimum inter-arrival times. However, as we demonstrate next with an example, in order to calculate worst-case response time exactly, inter-arrival and inter-departure times should be treated separately and there should be separate constraints. While putting a limit to the number of consecutive oc-

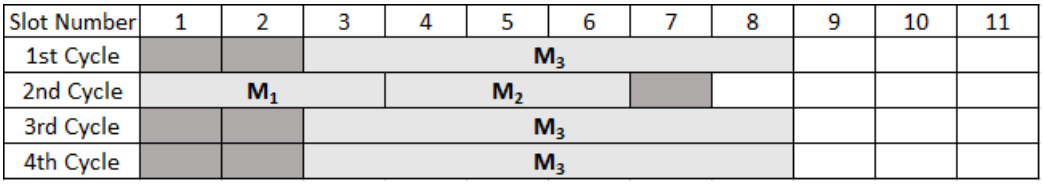

Figure 4.1: Worst-case Response Time Computation - Existing Algorithm

currences the constraint 4.6 does not account for the non-transmissibility of messages due to filled cycles. A message may be transmitted more times than it is generated within a certain number of cycles due to accumulated blocked instances.

We illustrate the described situation using the example presented in [12]. There are four sporadic messages m_1, m_2, m_3 and m_4. The number of minislots in the dynamic segment is 11 and the cycle time is T_c. The message sizes and minimum inter-arrival times are given in the Table 4.1.

Table4.1:Message Parameters

	Message Size in MS	Minimum Inter-Arrival Time in T_c	Deadline in T_c
m_1	3	4	4
m_2	3	4	4
m_3	6	1.9	1.9
m_4	3	5.5	5.5

If the WCRT for m_4 is computed using the presented algorithm, the resulting number of cycles that the message m_4 is blocked due to filled cycles, is $f = 4$. The messages transmitted may be seen in Figure 4.1.

According to the constraint 4.6, in 3 consecutive cycles, m_3 may be transmitted 2 times: $\lceil 3 \cdot T_c / 1.9 \cdot T_c \rceil = 2$. However, m_3 may be blocked by higher priority messages m_1 and m_2 and due to this blockage it may be possible that m_3 is transmitted 3 times within 3 consecutive cycles. It means that the inter-arrival and inter-departure counts of messages should be considered separately. If the first arrival of m_3 is at the initial cycle and just after the 3rd frame and if m_1 and m_2 are transmitted in the first cycle, then m_3 may be transmitted only in the 2nd, 3rd and 4th cycle. This situation is depicted in Figure 4.2.

Slot Number	1	2	3	4	5	6	7	8	9	10	11
1st Cycle	M_1			M_2							
2nd Cycle			M_3								
3rd Cycle			M_3								
4th Cycle			M_3								
5th Cycle	M_1			M_2							
6th Cycle			M_3								

Figure 4.2: Worst-case Response Time Computation - Blocking Effect Evaluated

When the blocking effect is accounted for in the analysis of this example, the number of cycles that message m_4 is blocked due to filled cycles, becomes 6. In order to compute the analysis with considering the blocking effect, we make a modification to the WCRT algorithm which is proposed in Section 4.2.2.

4.2.2 Improved Worst-Case Response Time Analysis

In this part we present the improved WCRT analysis algorithm. As it is mentioned in Section 4.2, inter-arrival and inter-departure counts of messages should be considered separately. In the existing algorithm, the constraint (4.6) behaves as if these two parameters are equal. We next introduce separate equations for the number of generations and number of transmissions. The maximum number of transmissions that a message D' has in j cycles where $1 \leq j \leq f$, may be evaluated as follows using the previously introduced boolean decision variables $x_{D',i}$:

$$\sum_{i=1}^{j} x_{D',i}. \tag{4.12}$$

It must hold for each cycle j, $1 \leq j \leq f$, that the number of transmissions of D' must be less or equal than the number of generations of D'. The number of transmissions is already calculated above. Moreover, the number of generations of D' until cycle j, $1 \leq j \leq f$, is computed as follows. We write

$$S_{D',j} := \sum_{\substack{D'' \\ a_{\mathrm{DS}}(D'') < a_{\mathrm{DS}}(D')}} x_{D'',j} \cdot N_{D'',\mathrm{MS}} \cdot T_{\mathrm{MS}} + (1 - x_{D''}) \cdot T_{\mathrm{MS}} \tag{4.13}$$

for the transmission duration of higher-priority messages D'' than D' in cycle j before the first time D' could be transmitted in that cycle. We further assume that the first instance of D' is generated right after its FID in cycle 0. That is, the message spends time $T_{\mathrm{DS}} - a_{\mathrm{DS}}(D') \cdot T_{\mathrm{MS}}$ in cycle 0, $(j-1) \cdot T_{\mathrm{c}}$ until cycle j and time $T_{\mathrm{c}} - T_{\mathrm{DS}} + S_{D',j}$ in cycle j. In addition, D' can be generated with a maximum frequency of $1/p_{D'}$ during that time. That is, the maximum number of generations until a potential transmission in cycle j is given by

$$\left\lceil \frac{T_{\mathrm{DS}} - a_{\mathrm{DS}}(D') \cdot T_{\mathrm{MS}} + (j-1) \cdot T_{\mathrm{c}} + T_{\mathrm{c}} - T_{\mathrm{DS}} + S_{D',j}}{p_{D'}} \right\rceil. \tag{4.14}$$

After several simplifications (4.14) becomes:

$$\left\lceil \frac{S_{D',j} + j \cdot T_{\mathrm{c}} - a_{\mathrm{DS}}(D') \cdot T_{\mathrm{MS}}}{p_{D'}} \right\rceil. \tag{4.15}$$

Hence, the constraint for message D' in cycle j that relates the number of generations until that cycle and the number of transmissions becomes:

$$\sum_{i=1}^{j} x_{D',i} \leq \left\lceil \frac{S_{D',j} + j \cdot T_{\mathrm{c}} - a_{\mathrm{DS}}(D') \cdot T_{\mathrm{MS}}}{p_{D'}} \right\rceil. \tag{4.16}$$

with $S_{D',j}$ in (4.13).

The new constraint (4.16) needs to be added to the LIP formulation. Unfortunately, the LIP formulation does not allow the usage of the ceiling operation. Accordingly, the constraint (4.16) should be transformed to a version that models the ceiling operation by linear expressions. Indeed, (4.14) can be expressed as follows:

$$\sum_{\substack{D'' \\ a_{\mathrm{DS}}(D'') < a_{\mathrm{DS}}(D')}} x_{D'',j} \cdot N_{D'',\mathrm{MS}} \cdot T_{\mathrm{MS}} + (1 - x_{D''}) \cdot T_{\mathrm{MS}} + j \cdot T_{\mathrm{c}} - a_{\mathrm{DS}}(D') \cdot T_{\mathrm{MS}} + K_{D',j} = c_{D',j} \cdot p_{D'}. \tag{4.17}$$

Here, a dummy integer constant $K_{D',j}$ and a dummy integer coefficient $c_{D',j}$ are added for each D' and j to (4.14) such that (4.17) is obtained. The equation reflects that, after performing the ceiling operation, the result should be an integer multiple of $p_{D'}$ and $K_{D',j}$ is the required remainder. Hereby, we want to use the smallest value for $K_{D',j}$ that fulfills the above equation in order to realize the ceiling function. That is,

it is sufficient to set boundaries of $K_{D',j}$ as $0 \leq K_{D',j} \leq p_{D'}$ such that $K_{D',j}$ can assume only one suitable value. Then, $c_{D',j}$ represents the result of the ceiling function in (4.16).

In summary, the improved WCRT analysis for a given message D evaluates

$$J^*_{D,f} = \max_X J_{D,f} \tag{4.18}$$

subject to the constraints (4.7), (4.8) and for all D' with $(D') < a(D)$ and all $1 \leq j \leq f$

$$\sum_{i=1}^{j} x_{D',i} \leq c_{D',j}, a$$

$$j \cdot T_{\mathrm{c}} - a_{\mathrm{DS}}(D') \cdot T_{\mathrm{MS}} + \sum_{\substack{D'' \\ a_{\mathrm{DS}}(D'') < a_{\mathrm{DS}}(D')}} x_{D'',j} \cdot N_{D'',\mathrm{MS}} \cdot T_{\mathrm{MS}} + (1 - x_{D''}) \cdot T_{\mathrm{MS}} + K_{D',j} = c_{D',j} \cdot p_{D'},$$

$$0 \leq K_{D',j} < p_{D'}.$$

If the message D may be transmitted in cycle f, then there should be room for the message D in the cycle: $J^*_{D,f} \leq (N_{\mathrm{latest}} - 1) \cdot T_{\mathrm{MS}}$ should be satisfied. If this condition is fulfilled, the total delay that message D experiences is:

$$w_D = t_{D,\mathrm{init}} + (f-1) \cdot T_{\mathrm{c}} + T_{\mathrm{SS}} + J^*_{D,f} + N_{D,\mathrm{MS}} \cdot T_{\mathrm{MS}}. \tag{4.19}$$

4.2.3 Execution and Case Study

In this section we present several test cases to which we apply our algorithm. Firstly, the case in (4.2) is revisited. We call this case Case 1. The message parameters are given in Table 4.1. We assume $T_{\mathrm{c}} = 20\,\mathrm{ms}$, $N_{MS} = 11$, $T_{\mathrm{DS}} = 11\,\mathrm{ms}$, $T_{\mathrm{SS}} = 7\,\mathrm{ms}$, $T_{\mathrm{SW}} + T_{\mathrm{NIT}} = 2\,\mathrm{ms}$. For this case, while computing the WCRT w_4 for m_4, the number of cycles that must be checked $f = \lceil \frac{5.5 \cdot T_c}{T_c} \rceil = 6$. When the analysis is executed for this case the result is as shown in Fig. 4.3.

From Figure 4.3, it may be observed that m_4 misses its deadline with a WCRT of $w_4 = 130\,\mathrm{ms}$. Fig. 4.4 shows the cycles in which m_4 is blocked.

The resulting blocked cycles in Figure 4.4 shows that the change of previous constraint on consecutive occurrences 4.6 to a constraint that take into consideration the

Messages	Deadline (ms)	Worst-Case Response Time (ms)	Deadline Met
m_1	80	23	✓
m_2	80	25	✓
m_3	38	46	×
m_4	110	130	×

Figure 4.3: The Result of the Case 1, $N_{MS} = 11$

Slot Number	1	2	3	4	5	6	7	8	9	10	11
1st Cycle	M_1			M_2							
2nd Cycle			M_3								
3rd Cycle			M_3								
4th Cycle			M_3								
5th Cycle	M_1			M_2							
6th Cycle			M_3								

Figure 4.4: Blocked Cycles in Case 1, $N_{MS} = 11$

inter-arrival and inter-departure times 4.16 works correctly.

In order for m_4 to be schedulable without missing its deadline, we change the FlexRay parameters N_{MS}, T_{DS}, *and* T_{SS} such that $N_{\mathrm{MS}} = 13$, $T_{\mathrm{DS}} = 13\,\mathrm{ms}$, $T_{\mathrm{SS}} = 5\,\mathrm{ms}$. T_{c} and $T_{\mathrm{SW}} + T_{\mathrm{NIT}}$ remain the same as before. When the analysis is performed with these modified parameters, the results is as in Figure 4.5.

According to the results of Figure 4.5, m_4 may be scheduled without missing its deadline. Figure 4.6 shows the distribution of messages to cycles. In this configuration the latest slot that a message may start transmission is evaluated as $N_{Latest} = 13 - 6 + 1 = 8$ ref. m_4 is blocked at the $1st$ and $2nd$ cycle by m_3, but, at cycle 3, m_1 and m_2 cannot block m_4 since N_{Latest} can not be reached. The WCRT of m_4 is evaluated as

Messages	Deadline (ms)	Worst-Case Response Time (ms)	Deadline Met
m_1	80	23	✓
m_2	80	25	✓
m_3	38	46	×
m_4	110	67	✓

Figure 4.5: Blocked Cycles Case 1, 13 *MS*

Slot Number	1	2	3	4	5	6	7	8	9	10	11	12	13
1st Cycle			M_3										
2nd Cycle			M_3										
3th Cycle	M_1			M_2				M_4					

Figure 4.6: The Result of the Case 1, 13 *MS*

67 ms. The initial delay according to 4.4 is evaluated as $t_{D,\text{init}} = (13-4+1)\cdot 1\,\text{ms} + 2\,\text{ms} = 12ms$. Since it is blocked by higher priority messages in 2 cycles, a delay of $2\cdot T_\text{c} = 2\cdot 20\,\text{ms} = 40\,\text{ms}$ should be added to the total delay. In cycle 3, in which m_4 is transmitted, a final delay is experienced. This final delay is the time passed until m_4 is transmitted. The time passed in the static segment is $T_\text{SS} = 5\,\text{ms}$ and the transmission of m_4 is finished at the $10th$ slot, after 10 ms from the beginning of the DS. Therefore, the final delay evaluates to $5\,\text{ms} + 10\,\text{ms} = 15\,\text{ms}$ and the total delay is $12\,\text{ms} + 40\,\text{ms} + 15\,\text{ms} = 67\,\text{ms}$. Note that m_3 still misses its deadline. N_MS should be made larger for m_3 not to miss its deadline.

We perform a second test case with 10 messages, Case 2. We choose $T_\text{c} = 30\,\text{ms}$, $N_\text{MS} = 20$, $T_\text{DS} = 20\,\text{ms}$, $T_\text{SS} = 8\,\text{ms}$, $T_\text{SW} + T_\text{NIT} = 2\,\text{ms}$. The sizes and minimum inter-arrival times of messages are presented in Table 4.2.

Table4.2:Message Parameters of Case 2

	Message Size in MS	Minimum Inter-Arrival Time in *ms*	Deadline
m_1	3	300	100
m_2	3	120	60
m_3	4	75	50
m_4	4	105	60
m_5	3	150	90
m_6	3	180	90
m_7	3	90	70
m_8	5	90	75
m_9	4	90	75
m_{10}	3	120	105

The result of the analysis for Case 2 is presented in Figure 4.7. It may be seen that two messages miss their deadlines, m_9 and m_{10}. In order to schedule all the messages in this set without changing the FIDs assigned, the number of minislots in the DS should be increased. We re-perform the analysis with the following parameters: $T_\text{c} =$

Messages	Deadline (ms)	Worst-Case Response Time (ms)	Deadline Met
m_1	100	33	✓
m_2	60	35	✓
m_3	50	38	✓
m_4	60	41	✓
m_5	90	43	✓
m_6	90	63	✓
m_7	70	67	✓
m_8	75	72	✓
m_9	75	104	×
m_{10}	105	133	×

Figure 4.7: The Result of the Case 2, 20 *MS*

30 ms, $N_{\mathrm{MS}} = 25$, $T_{\mathrm{DS}} = 25\,\mathrm{ms}$, $T_{\mathrm{SS}} = 3\,\mathrm{ms}$, $T_{\mathrm{SW}} + T_{\mathrm{NIT}} = 2\,\mathrm{ms}$. The result may be observed in Figure 4.8. When the slot number N_{MS} is increased from 20 to 25 without altering the cycle time T_{c} by decreasing T_{SS} to 3 ms, m_9 and m_{10} become schedulable.

We also apply our algorithm to the benchmark problem in [12] and compare the results. In [12], they use the benchmark of the Society of Automotive Engineers (SAE) [1] which provides a set of signals for a prototype electric car. In total, there are 53 signals and 31 of them are sporadic. Since periodic messages are scheduled in the static segment, we only consider the sporadic messages. In [12], they use three different configurations for the FlexRay network. Table 4.3 shows the configuration parameters.

Table4.3:FlexRay Configurations of [12]

Configuration	$T_c\ (MT)$	$T_{SS}\ (MT)$	$T_{MS}\ (MT)$	N_{MS}
conf.1	170	60	2	50
conf.2	120	40	2	40
conf.3	150	30	2	60

We present the sporadic message parameters for the three different configurations at Table 4.4 and computed the WCRT analysis for all messages under these different configurations. Our computation does not return in a reasonable time (1 hour) for the messages 20-31 under configurations 1 & 2 and messages 20 & 22-31 under configu-

Messages	Deadline(ms)	Worst-Case Response Time (ms)	Deadline Met
m_1	100	33	✓
m_2	60	35	✓
m_3	50	38	✓
m_4	60	41	✓
m_5	90	43	✓
m_6	90	63	✓
m_7	70	47	✓
m_8	75	67	✓
m_9	75	71	✓
m_{10}	105	74	✓

Figure 4.8: The Result of the Case 2, 25 *MS*

ration 3. We confirmed that we have the same results of schedulability analysis with the algorithm of [12] for the messages 1-19 under configurations 1 & 2 and messages 1-19 & 21 under configuration 3. Since we use an open source non-professional LIP solver while implementing our algorithm, we encounter cases for which our algorithm does not return in a reasonable time.

Table4.4:Sporadic Message Parameters [12]

Message ID	Message Size (MS)	Deadline conf.1 (MT)	Deadline conf.2 (MT)	Deadline conf.3 (MT)
1	7	1000	720	1400
2	4	800	720	1400
3	4	800	720	1400
4	5	800	720	1400
5	4	900	720	1400
6	4	800	850	1400
7	6	800	850	1400
8	6	800	850	1400
9	4	1000	1500	1400
10	4	1000	1500	1400
11	4	1000	1500	1400
12	4	800	1500	1400
13	4	800	1500	1400
14	4	1000	1500	1400
15	5	700	1500	1400
16	11	1000	1500	1400
17	4	1000	1500	1400
18	4	900	1500	1400
19	4	900	1500	1400
20	10	1200	1500	1400
21	4	2000	1500	1400
22	4	2000	1500	1400
23	4	2000	1500	1400
24	4	2000	1500	1400
25	4	2000	1500	1400
26	4	20000	1500	1400
27	4	2000	1500	1400
28	5	2000	1500	1400
29	4	2000	1500	1400
30	11	2000	1500	1400
31	4	2000	1600	1400

4.3 Dynamic Segment Scheduling

In this section, the scheduling problem on the FlexRay DS is investigated. Similar to the study of the WCRT, our study in this section depends on the study of [18]. The scheduling of dynamic segment messages has two components: assigning an FID to each message such that no message misses its deadline and setting the number of slots N_{MS} in the DS. The problem of scheduling is computationally tedious since the number of possible assignment of FIDs and N_{MS} is large. If there are $|\mathcal{D}|$ messages to be scheduled, then there are $\begin{pmatrix} N_{\mathrm{MS}} \\ |\mathcal{D}| \end{pmatrix} \cdot |\mathcal{D}|!$ possible FID assignments. If we accept that the first $|\mathcal{D}|$ minislots are assigned as dynamic slots then there are $|\mathcal{D}|!$ possible FID assignments.

input : Message set $\mathcal{D}$, $N_{\mathrm{MS}} = N_{\mathrm{MS,max}}$, $N_{\mathrm{MS}} = \max_{D \in \mathcal{D}} N_{D,\mathrm{MS}}$, a_{DS} is empty.
while($N_{\mathrm{MS}} \leq N_{\mathrm{MS,max}}$) 1
  $i = x$, $\mathcal{D}' = \mathcal{D}$ 2
  while $\mathcal{D}' \neq \emptyset$ 3
    $i = i + 1$ 4
    schedulable = **true** 5
    for each message $D \in \mathcal{D}'$ 6
      set $a_{\mathrm{DS}}(D) = i$ 7
      if D is unschedulable according to (4.19) 8
        schedulable = **false** 9
        break 10
      else 11
        record $d_D - w_D$ 12
    if *schedulable* = **false** 13
      $N_{\mathrm{MS}} = N_{\mathrm{MS}} + 1$ 14
      break 15
    set $a_{\mathrm{DS}}(D) = i$ for $D \in \mathcal{D}$ with minimal $d_D - w_D$ 16
    $\mathcal{D}' = \mathcal{D}' - \{D\}$ 17
    if $\mathcal{D}' = \emptyset$ 18
      return a_{DS} 19
return false 20

Algorithm 5: Scheduling algorithm for the FlexRay DS in [18].

In [18], the heuristic Algorithm 5 is presented. The algorithm iteratively performs schedulability analysis for a candidate value N_{MS}. If schedulability analysis fails then the value of N_{MS} is incremented by 1 and the analysis starts from the beginning. The starting value of the N_{MS} equals the largest message size in minislots. The algorithm

assigns FIDs in increasing order to the messages which have the smallest difference between the deadline and worst-case response time. The problem we point out for the algorithm is its run-time. The run-time depends on the number of messages and the characteristics of message parameters. As the message number gets larger, the run-time of the algorithm increases exponentially.

In order to resolve the problem of a large run-time, we propose a heuristic algorithm, Algorithm 6, which is simpler than the algorithm in [18]. It is based on the observation that Algorithm 5 generally assigns the FIDs in the DS with increasing message deadline. Accordingly, we propose to assign FIDs to messages according to their deadlines; the message with smaller deadline gets a smaller FID. There are two issues with this FID assignment scheme:

1. Messages with identical deadlines and different periods: We assign higher-priority FIDs to messages with smaller periods.

2. Messages with identical deadlines and different sizes: We assign higher-priority FIDs to messages with larger sizes.

At the beginning of algorithm, we assign FIDs to all messages according to the described method. Then, we apply a bisection algorithm, that is, N_{MS} is set to the half of the sum of $N_{MS_{max}}$ and $N_{MS_{min}}$ values. $N_{MS_{max}}$ is the sum of all message sizes and $N_{MS_{min}}$ is the longest message size. The WCRT of all messages is analyzed using the algorithm of Section 4.2.2. If at least one message misses its deadline with the assigned N_{MS}, N_{MS} is set to the average of N_{MS} and $N_{MS_{max}}$. If all messages meet their deadlines, N_{MS} is set to the average of N_{MS} and $N_{MS_{min}}$ and the analysis procedure starts over again. The algorithm is terminated if the value of N_{MS} does not change.

The algorithm is applied to the message set in Table 4.5. Static segment duration is set to 7 ms and the total duration of SW and NIT is set to 2 ms for this case.

The resulting FID assignment that is computed by heuristic algorithm is given in Table 4.6 and the number of MSs in dynamic segment N_{MS} is determined as 11. The algorithm of [18] results in the same FID assignment and N_{MS} as our heuristic scheduling algorithm for this case.

input : Message Set $\mathcal{D}$

output: FID assignment for $\mathcal{D}$

Assign FIDs in increasing order of deadlines
$N_{\mathrm{MS}_{\max}} = sum\ of\ all\ message\ sizes$
$N_{\mathrm{MS}_{\min}} = \max_{D \in \mathcal{D}} N_{D,\mathrm{MS}}$
$N_{\mathrm{MS}} = (N_{\mathrm{MS}_{\max}} + N_{\mathrm{MS}_{\min}})/2$
while !*terminate* **do**
    for *each* $D \in \mathcal{D}$ **do**
        success = `ComputeWCRTAnalysis` for D
        if *success* **then**
            $N_{\mathrm{MS}} = (N_{\mathrm{MS}} + N_{\mathrm{MS}_{\min}})/2$
        else
            $N_{\mathrm{MS}} = (N_{\mathrm{MS}} + N_{\mathrm{MS}_{\max}})/2$
        end
    end
    if N_{MS} *does not change* **then**
        terminate
    end
end

Algorithm 6: Heuristic DS Scheduling Algorithm

In order to evaluate the run time performance of our heuristic algorithm, we randomly generate test cases and run our heuristic algorithm and algorithm of [18] on these test cases and compare run times. We generate random message sets which have 5 and 10 messages. We set T_{SS} to 5 ms and minimum inter-arrival times are chosen between $90 - 190$ ms. We generate 10 random cases for each message count. We present the average run-times at Table 4.7. We should note that for all the cases, the two algorithms result in same FID assignment and same N_{MS}.

It may be seen from Table 4.7 that our heuristic algorithm has a similar run-time for the 5 message case and has a lower run-time for the 10 message case than the algorithm of [18]. We should note that both algorithms do not return in a reasonable time for the cases with 20 or higher number of messages. This is due to the usage of a non-professional LIP solver used in our implementation.

Table4.5:Message Parameters

	Message Size in MS	Minimum Inter-Arrival Time in ms	Deadline in ms
m_1	8	150	100
m_2	4	130	90
m_3	6	200	150
m_4	4	120	120

Table4.6:FID Assignment

	FID Assigned
m_1	2
m_2	1
m_3	4
m_4	3

Table4.7:Dynamic Segment Message Scheduling Algorithm Run-Time Evaluation

Message Count	Average Run-Time of [18] in s	Average Run-Time of our heuristic in s
5	0.063	0.067
10	5.06	0.78

CHAPTER 5

CONCLUSION

With the recent advances in the automotive industry, the number of electronic components in automobiles considerably increased. Alongside the rise in the electronic components, the amount of data exchanged in the automobiles has increased. As a result, scheduling of messages that carry this data became more complex. The ECUs used in safety and X-by-Wire applications made it very critical not to miss message deadlines.

In this respect, we focus on the schedule analysis and design problems of both Static (SS) and Dynamic Segments (DS) of the FlexRay protocol. Regarding the SS, we propose a new heuristic approach for the FID assignment that avoids the LIP solvers which cannot be practically integrated into other software needed for the optimal result. Regarding the DS, the thesis proposes a WCRT analysis that also produces correct results for infeasible priority assignments, different than the previous work. Furthermore, a low complexity priority assignment for the DS is proposed.

All proposed approaches are tested and evaluated by case studies. The results show that the heuristic SS scheduling approach achieves close results to the optimal assignment defined by the objective function with weights for the efficient use of FIDs and jitter. The DS WCRT analysis provides correct results for infeasible schedules as well. The DS scheduling algorithm runs faster with respect to the previous approaches without compromising the WCRT.

It has to be noted that the FlexRay standard continues to evolve. The newly published FlexRay Communications System Protocol Specification Version 3.0.1 has additional

features including more relaxed FID assignment constraints in the SS. Such changes redefine the scheduling problem. We believe that the approaches proposed in this thesis can be adopted to the new standard maintaining their good properties.

REFERENCES

[1] Class C application requirement considerations. Technical Report J2056/1, Society for Automotive Engineers, 1993.

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

[3] H. Kopetz. A solution to an automotive control system benchmark. In *IEEE Real-time systems symposium*, 1994.

[4] H. Kopetz and G. Bauer. The time-triggered architecture. *Proc. IEEE*, 91(1):112–126, 2003.

[5] R. Lange, F. Vasques, P. Portugal, and R. De Oliveira. Guaranteeing real-time message deadlines in the flexray static segment using a on-line scheduling approach. In *IEEE International Workshop on Factory Communication Systems - Proceedings, WFCS*, pages 301–310, 2012.

[6] M. Lukasiewycz, M. Glaß, J. Teich, and P. Milbredt. FlexRay schedule optimization of the static segment. In *IEEE/ACM international conference on Hardware/software codesign and system synthesis*, pages 363–372, 2009.

[7] M. Lukasiewycz, R. Schneider, D. Goswami, and S. Chakraborty. Modular scheduling of distributed heterogeneous time-triggered automotive systems. In *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, pages 665–670, 2012.

[8] R. Makowitz and C. Temple. FlexRay - a communication network for automotive control systems. *Factory Communication Systems, IEEE International Workshop on*, pages 207–212, June 27, 2006.

[9] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert. Trends in automotive communication systems. *Proceedings of the IEEE*, 93(6):1204–1224, 2005.

[10] 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*, 46:607 – 617, 2000.

[11] M. Neukirchner, M. Negrean, R. Ernst, and T. Bone. Response-time analysis of the flexray dynamic segment under consideration of slot-multiplexing. In *7th IEEE International Symposium on Industrial Embedded Systems, SIES 2012 - Conference Proceedings*, pages 21–30, 2012.

[12] L. Ouedraogo and R. Kumar. Computation of the precise worst-case response time of flexray dynamic messages. *IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING*, 11(2):537–548, 2014.

[13] I. Park and M. Sunwoo. Flexray network parameter optimization method for automotive applications. *IEEE Transactions on Industrial Electronics*, 58(4):1449–1459, 2011.

[14] T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei. Timing analysis of the FlexRay communication protocol. *Real-Time Syst.*, 39(1-3):205–235, 2008.

[15] E. G. Schmidt and K. Schmidt. Message scheduling for the FlexRay protocol: The dynamic segment. *Vehicular Technology, IEEE Transactions on*, 58(5):2170–2179, 2009.

[16] K. Schmidt and E. G. Schmidt. Message scheduling for the FlexRay protocol: The static segment. *Vehicular Technology, IEEE Transactions on*, 58(5):2160–2169, 2009.

[17] K. Schmidt and E. G. Schmidt. Optimal message scheduling for the static segment of FlexRay. In *Vehicular Technology Conference Fall*, pages 1 –5, sep. 2010.

[18] K. Schmidt and E. G. Schmidt. Schedulability analysis and message schedule computation for the dynamic segment of FlexRay. In *Vehicular Technology Conference Fall*, pages 1 –5, sep. 2010.

[19] I. Standard-11898. Road vehicles-interchange of digital information – Controller Area Network (CAN) for high-speed communication. *International Standards Organisation (ISO)*, 1993.

[20] B. Tanasa, U. Bordoloi, S. Kosuch, P. Eles, and Z. Peng. Schedulability analysis for the dynamic segment of flexray: A generalization to slot multiplexing. In *Real-Time Technology and Applications - Proceedings*, pages 185–194, 2012.

[21] Y. Xie, G. Zeng, Y. Chen, R. Kurachi, H. Takada, and R. Li. Worst case response time analysis for messages in controller area network with gateway. *IEICE Transactions on Information and Systems*, 2013(7):1467–1477, 2013.

[22] H. Zeng, A. Ghosal, and M. Di Natale. Timing analysis and optimization of flexray dynamic segment. In *Proceedings - 10th IEEE International Conference on Computer and Information Technology, CIT-2010, 7th IEEE International Conference on Embedded Software and Systems, ICESS-2010, ScalCom-2010*, pages 1932–1939, 2010.