CONTROLLER AREA NETWORK RESPONSE TIME ANALYSIS AND SCHEDULING FOR ADVANCED TOPICS: OFFSETS, FIFO QUEUES AND GATEWAYS

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

BY

BURAK ALKAN

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

FEBRUARY 2015

Approval of the thesis:

CONTROLLER AREA NETWORK RESPONSE TIME ANALYSIS AND SCHEDULING FOR ADVANCED TOPICS: OFFSETS, FIFO QUEUES AND GATEWAYS

submitted by **BURAK ALKAN** 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 Dept., METU** _____

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

Examining Committee Members:

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

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

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

Utku Karakaya, M.Sc.
TOFAS ARGE _____

Vakkas Çelik, M.Sc.
TUBITAK UZAY _____

Date: **10.2.2015**

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: BURAK ALKAN

Signature :

ABSTRACT

CONTROLLER AREA NETWORK RESPONSE TIME ANALYSIS AND SCHEDULING FOR ADVANCED TOPICS: OFFSETS, FIFO QUEUES AND GATEWAYS

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

February 2015, 112 pages

Controller Area Network (CAN) is the most widely used in-vehicle network for the communication among electronic control units (ECUs). CAN has a priority-based arbitration mechanism and the classical usage of CAN assumes the implementation of priority queues (PQs) on ECUs. Based on this assumption, the literature provides efficient algorithms for the computation of worst-case response times (WCRTs) of messages as well as for the appropriate assignment of priorities to messages in order to meet real-time guarantees such as message deadlines.

In contemporary CAN networks there are several extensions to the classical case. First, the addition of new functionality to vehicles requires adding new messages with appropriate priorities to existing CAN networks. Second, FIFO queues (FQs) might be used instead of PQs for easier implementation. Third, due to the ever-increasing bus load, CAN networks are usually divided into several segments that are connected via *gateways* to decrease the contention among messages. Fourth, a further measure is to distribute the message transmission of each ECU over time by assigning

transmission *offsets* to messages. All of the stated extensions require new methods for WCRT analysis and priority assignment on CAN.

This thesis has a list of contributions that address the extensions for CAN as listed above. Regarding offset scheduling; different schedulability analysis methods for message sets with given offset and priority assignments are incorporated to a previous offset assignment algorithm. Then, a new algorithm which simultaneously assigns the message offsets and priorities is proposed. Regarding ECUs with FIFO queues; the previous schedulability analysis is improved to decrease its run time and then this analysis is used in an algorithm that assigns the priorities to the new messages that extend an existing CAN network. Regarding gateways; an algorithmic priority assignment is proposed for ECUs with priority queues and the schedulability analysis for CAN networks with gateways is extended to FIFO queues.

All of the algorithms that are used and developed in this thesis are implemented in C++ to integrate into a novel in-vehicle network analysis and design tool; AUTONET.

Keywords: Controller Area Network (CAN), Schedulability analysis, Response time analysis, Priority assignment, CAN FIFO Analysis, CAN Offset Analysis, CAN-to-CAN Gateway Analysis

ÖZ

DENETLEYİCİ ALAN AĞI(DAA) İLERİ KONULARI İÇİN TEPKİ SÜRESİ ANALİZİ VE ÇİZELGELENDİRME: GÖRELİ KONUM, FIFO KUYRUKLAR VE AĞ GEÇİTLERİ

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

Şubat 2015 , 112 sayfa

DAA elektronik kontrol üniteleri (EKÜ) arasındaki haberleşme en yaygın olarak kullanılan araç içi haberleşme ağıdır. DAA'in öncelik tabanlı bir tahkim mekanizması vardır ve klasik kullanımda EKÜ'ler üzerinde öncelik kuyrukları kullanıldığı varsayılır. Bu varsayıma dayanarak, literatürde en kötü durum tepki süreleri hesaplamaları ve mesajların gerçek zamanlı zaman sınırlarını karşılayan uygun öncelik atamaları için etkili algoritmalar bulunmaktadır.

Aktüel DAA ağlarında klasik duruma ek olarak başka durumlar vardır. Birincisi, araçlara yeni özelliklerin eklenmesi, mevcut DAA ağlarına uygun öncelikleri ile yeni mesajların eklenmesini gerektirir. İkincisi, FIFO kuyrukları çeşitli nedenlerden dolayı öncelik kuyruklarının yerine kullanılabilir. Üçüncü olarak, sürekli artan veri yoğunluğundan dolayı DAA ağları, ağ geçitleriyle mesajlar arasındaki çekişmeyi azaltmak için çeşitli kısımlara ayrılmıştır. Dördüncü olarak diğer bir önlem mesajlara göreceli konumlar atayarak mesaj iletimini zamana yaymaktır. Belirtilen tüm bu ek özellikler

tepki süresi hesaplamaları ve öncelik atamaları için yeni metotları gerektirir.

Bu tez yukarıda listelenen DAA ileri konuları için yapılan katkıları bir liste olarak sunmaktadır. Göreceli konum çizelgelemesi için, göreceli konum ve öncelik değerleri belirlenmiş mesaj setleri için farklı çizelgeleme analiz yöntemleri daha önceki göreceli konum atama algoritmalarıyla birleştirilmiştir. Sonra, aynı anda öncelik ve göreceli konum ataması yapabilen yeni bir algoritma önerilmektedir. FIFO kuyruğu kullanan EKÜ'ler için, daha önceki çizelgeleme analiz yöntemi geliştirilmiş ve bu algoritmanın çalışma süresi düşürülmüştür. Daha sonra bu algoritma var olan bir mesaj setine ekleme yapabilmek için öncelik atayabilecek hale getirilmiştir. Ağ geçitleri için, öncelik kuyruğu kullanan EKÜ'lere yönelik öncelik atama algoritması önerilmiş ve halihazırdaki çizelgeleme analizi FIFO kuyrukları da kapsayacak şekilde geliştirilmiştir.

Bu tezde kullanılan ve geliştirilen tüm algoritmalar C++'da yazılmış ve yeni bir araç içi ağ analiz ve tasarım aracı olan AUTONET'e entegre edilmiştir.

Anahtar Kelimeler: Denetleyici Alan Ağı(DAA), Çizelgeleme Analizi, Tepki Süresi Analizi, Öncelik Atamaları, DAA FIFO Analizi, DAA Gecikme Analizi, DAA Ağ Geçidi Analizi

To My Wife

ACKNOWLEDGMENTS

First of all, I would like to express my sincere appreciations to my supervisors Assoc. Prof. Dr. Şenan Ece Güran Schmidt and Assoc. Prof. Dr. Klaus Schmidt for their excellent guidance and insight throughout this thesis. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better advisors and mentors than them for my study.

I would like to thank TÜBİTAK-BİDEB for their financial support during my graduate education. I am supported by TÜBİTAK-BİDEB MS scholarship (2210). I would also like to express my gratitude to Utku Karakaya, Duygu Çulum Karani and Onur Canbaz from TOFAS R&D team for their contribution to my research.

A special thanks to my wife, Emine. Words cannot express how grateful I am to my wife for all of the sacrifices that she made on my behalf. Her support was what sustained me thus far.

I want to thank to ASELSAN Inc. and my employers for letting me involve in this academic study. I would like to express my special appreciation to my colleagues Erman, Korhan, Fatih, Emine, Eray, Hakan and my seniors Emrah Demircan, Nurise Savaşır Özpolat for their contribution on the improvement of my engineering skills, which makes it possible to finish this thesis.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF ABBREVIATIONS

CAN	Controller Area Network
ECU	Electronic Control Unit
SOSA	Standard Offset Scheduling Algorithm
LNSA	Local Neighborhood Search Algorithm
GAOS	Genetic Algorithm for Offset Scheduling
CSMA/CR	Carrier Sense Multiple Access/Collision Resolution
GUI	Graphical User Interface
WCRT	Worst Case Response Time
ID	Identifier (Priority)
CAN2CAN	CAN-to-CAN
LCM	Least Common Multiple
WCEA	Worst Case Exploration Algorithm
MIF	Maximum Interference Function
IF	Interference Function
FIFO	First-in-first-out
PQ	Priority Queue
FQ	FIFO Queue
BCRT	Best Case Response Time
GW	Gateway
non-GW	non-Gateway
ADC	Absolute Distance Constraint
AutoNET	Automotive Network Designer

CHAPTER 1

INTRODUCTION

Controller Area Network (CAN) is the de-facto standard for the signal communication among electronic control units (ECU) in vehicles. It was developed in 1983 at Bosch as an internal project for in-vehicle networks. In 1986, CAN became official and CAN chips were released commercially. Standards [40] were published and CAN controllers are now used in millions of cars [43]. CAN supports bounded message delays by always sending the highest priority message among multiple messages in different ECUs that are contending for the bus access at the same time. In order to support message transmission on CAN with real-time guarantees, it is hence essential to determine their worst-case response times (WCRT) depending on the message properties such as message priority, size and period [10]. It is then necessary, that all messages are schedulable, that is, there WCRTs are smaller than their deadlines. In addition, assuming that message priorities are unassigned for a new in-vehicle application, it is as well desired to assign priorities such that each message meets its deadline.

The stated problems are addressed in the academic literature. Algorithms for the WCRT analysis are first stated in [41] and improved in [19]. Furthermore, many studies are concerned with the message priority assignment on CAN such as [14, 19, 22]. Hereby, the classical usage of CAN in a single CAN network with ideal CAN nodes that implement priority queues (PQs) for their generated CAN messages is assumed. Nevertheless, several modifications in the usage of CAN are made in practical applications which requires special attention.

First, it has to be considered that in-vehicle applications are not always developed

from scratch but evolve over time. That is, it is common in practical applications that existing message sets are extended by new messages when adding functionality. In that case, priorities need to be assigned to the new messages while keeping the IDs of the existing messages. This problem is studied in [39] under the assumption of the classical usage of CAN with PQs. Second, practical applications do not always use CAN nodes with PQs but use FIFO queues (FQs) instead for an easier implementation. CAN networks with FQs require a modified WCRT analysis. The reason is that not always the highest-priority message contents for the bus but might be waiting in an FQ behind a lower-priority message, increasing the WCRT. A WCRT analysis for CAN networks with FQs is proposed in [25] together with a priority assignment algorithm. Third, it is observed in recent practical applications that the bus load on CAN increases considerably due to new functionality that is added to vehicles. Since the classical usage of CAN poses a limit on the supported bus load, the transmission of messages on CAN with offsets is suggested [31]. Here, the idea is to avoid that messages of the same CAN node are transmitted simultaneously by assigning a unique offset to each message. As a result, the number of messages that contents for the bus at any time is reduced, which also reduces the WCRT of messages and hence allows for a higher bus load. Different methods for the WCRT analysis for CAN with offsets are provided in [26, 47, 17]. Fourth, it is common that in-vehicle applications do not contain only a single CAN network but multiple CAN networks. Since data exchange between these networks is usually required, CAN-CAN gateways are used in order to exchange messages between different CAN networks. Considering the dependency of CAN networks that are connected by a gateway, a modified WCRT analysis is suggested in [45].

It can be observed that the literature offers different approaches for the more advanced topics on CAN. Nevertheless, there are several missing points in the literature and the combination of the advanced topics has not been addressed at all. The main aim of this thesis is to solve some of the missing points in the literature supported by efficient algorithms. In particular, the contributions of this thesis are as follows:

- Several existing algorithms for the stated advanced topics are implemented in the form of a C++ software library. These algorithms include the WCRT analysis for CAN networks with FQs from [25, 11], different algorithms for the

WCRT analysis for CAN networks with offsets from [26, 47, 17], algorithms for the offset assignment on CAN networks with offsets from [16], and the WCRT analysis for gateway networks from [45]. The algorithmic implementation is accompanied by several case studies.

- Regarding CAN networks with offsets, the current literature only provides algorithms that assign offsets to messages, assuming that the message priorities are pre-assigned. As a novel contribution of this thesis, a new algorithm that assigns both offsets and message priorities is developed and implemented in the C++ software library. An experimental case study shows the effectiveness of this algorithm.

- Regarding CAN networks with FQs, the problem of extending an existing CAN network with FQs by new messages is considered and a new algorithm for the priority assignment of the new messages is proposed. The results of this research are also presented in [12].

- Regarding CAN gateway networks, only WCRT analysis is considered in the existing literature. This thesis develops an algorithm for the priority assignment on CAN gateway networks and demonstrates its effectiveness in computational experiments. In addition, the WCRT analysis on CAN gateway networks is extended to the case of FQs.

An overview of the relevant advanced topics and the related contributions of this thesis is given in Table 1.1.

Table1.1: Literature Search Results Table

No	Topic	Does Exist In the Literature?	This Thesis
1	Offset Analysis	Yes	Yes
2	Offset Assignment	Yes	Yes
3	Offset Assignment and Priority Assignment	No	Yes
4	FIFO Analysis	Yes	Yes
5	FIFO Schedule	Yes	Yes
6	FIFO Schedule with Message Set Extension	No	Yes
7	GW Analysis	Yes	Yes
8	GW Priority Assignment	No	Yes
9	FIFO-Offset Analysis	Yes	No
10	FIFO-Offset Assignment	No	No
11	FIFO-Offset Priority Assignment and Offset Assignment	No	No
12	FIFO-GW Analysis	Yes	Yes
13	FIFO-GW Priority Assignment	No	No
14	GW-Offset Analysis	No	No
15	GW-Offset Assignment	No	No
16	FIFO-Offset-GW Analysis	No	No
17	FIFO-Offset-GW Assignment	No	No

The remainder of the thesis is organized as follows. Chapter 2 gives the necessary background and motivation for the thesis work. Chapter 3 contains the thesis work on CAN networks with offsets. First, different WCRT analysis algorithms are implemented and compared: approximate, exact and MIF analysis. Then some offset assignment algorithms are examined and implemented using the three WCRT analysis algorithms. Their efficiencies are compared. In the final part of this chapter, offsets and priorities are assigned together in order to better utilize the CAN bus. In Chapter 4, the WCRT analysis for CAN with FQs is investigated and an improved algorithm is proposed. In addition, a method for extending existing message sets for CAN networks with FQs is developed. CAN-CAN gateways are studied in the Chapter 5. An existing gateway analysis technique is reviewed and implemented. Furthermore, a new gateway priority assignment algorithm is developed. Then by combining the results on CAN networks with FQs and gateways, a gateway analysis is conducted for CAN gateway networks with both FQs and PQs. Finally, Chapter 6 describes the integration of the described methods and algorithms into the software tool AutoNET.

CHAPTER 2

BACKGROUND AND MOTIVATION

2.1 CAN Protocol

CAN protocol was developed in 1983 in order to reduce the cabling weight in automobiles. Thanks to it, engine, brakes, air conditioner and all the car sensors are connected together with two cables instead of dozens of them. In automotive networks, CAN has the biggest market share and the most dominant product. In [6], it is stated that although it has been in use for 3 decades, it is still at the beginning of global market penetration. 850 million CAN related products were sold in 2011 and future predictions estimate growth for CAN protocol for the next two decades.

Although started to be used for automotive networks, because of some good reasons such as speed, robustness against errors and ease of implementation, now it is in use in many areas. Industrial automation, medical devices, railway transport, marine systems and the energy area are the other usage areas of CAN protocol.

CAN is an asynchronous multi-master serial data bus with Carrier Sense Multiple Access/Collision Resolution (CSMA/CR), that is capable of operating at speeds of up to 1 Mbit/s. Each CAN frame has a *payload* of at most 8 B and a unique *identifier* (CAN ID) such that the frame with the smallest ID has the highest bus access priority. Any CAN node may start a transmission when the bus is idle. If more than one node attempt transmission of a frame at the same time, the node whose frame has the CAN ID with the lowest binary value wins the arbitration. Other nodes that lose the arbitration retransmit their frames when the bus becomes idle again. In CAN hardware, receiver and transmitter are physically independent. The sent message

can be listened by the receiver at the same time. By this way, the CAN controller compares the signal it sent and it received. During transmission, ID is sent first. In the bus line, there are two logic levels: 0 and 1. 0 is dominant and 1 is recessive. If two different CAN controllers send their messages at the same time, collision occurs and it can be resolved by ANDing these logic levels. This is called bitwise arbitration. Message with lower ID becomes dominant and the other sending node decides to wait. Therefore, lower ID messages have higher priorities. When collision occurs, low priority message detects the collision immediately and waits until the line is empty. ID is also used for message filtering in the receiver part.

The arbitration mechanism relies on the fact that all of the messages have distinct CAN IDs and all nodes on a CAN network use the same type of CAN ID (11 bit for the base frame format and 29 bit for the extended format). Consider a CAN network with a bit time of τ_{bit}. It is possible to evaluate the *maximum transmission time* C of each CAN frame with a payload of b bytes as

$$C = (55 + 10b)\tau_{\text{bit}} \text{ (11-bit CAN ID) and } C = (80 + 10b)\tau_{\text{bit}} \text{ (29-bit CAN ID)} \quad (2.1)$$

In the sequel, we assume that all messages on a network have the same type of CAN ID and the respective maximum frame duration is given. We further note that both periodic and sporadic messages are subject to the same event-triggered arbitration mechanism.

There are several CAN standards, in CAN 2.0A standard, messages are packed into frames. There are 4 types of frames: Data frames, request frames, error frames and overload frames. These frames are used with different purposes: data, data request, faulty data and delayed data. CAN has multiple error detection and error handling mechanisms. In terms of reliability, it is a very robust protocol, but error handling is out of the scope of this thesis.

2.2 Problems of CAN Protocol

We have made literature search and found some problems that CAN protocol encounters. Some of these problems are already solved and some of them are still unresolved. In this thesis, we try to solve some of the open problems.

-Message Formation and Signal Packing: The goal of frame packing for CAN is packing different signals in the same message in order to reduce the framing overhead and hence improve the schedulability of CAN systems. In the most recent work, [29] packs signals with similar periods in the same frame such that the bandwidth utilization is reduced by each packed message. This measure is also beneficial for the schedulability of the CAN message set. In the scope of this thesis, it is assumed that signals are already packed into messages.

-Finding worst case response time bounds for messages: Since the car operations are real-time operations, they have to have some time deadlines. It should be guaranteed by network designers that message response times would not exceed their deadlines. For this purpose, we use analytical models to calculate message response times. Even if, we could not calculate it exactly, finding an upper bound works.

-Hardware limitations: Hardware limitations are limited number of buffers, non-abortable transmission requests and FIFO queues. Practical CAN applications rely on the functionality of the ECU hardware. In practise, several hardware components do not strictly follow the CAN protocol specification. Controller may have less number of buffers than the synchronously arrived messages. In such a case, exceeding messages wait in software queues and cannot take part in the arbitration if the transmission requests cannot be abortable which introduces additional delay. Instead of priority queues, sometimes FIFO queues are preferred, which is against the nature of CAN protocol. It causes priority inversion and messages may miss their deadlines.

-Finding an optimal priority ordering: We should give priorities to messages in a such a way that if a schedulable order exists for a given message set, it is guaranteed that we find it.

-Finding a robust priority ordering: Error due to electromagnetic interference is possible in CAN bus with a very small probability. Although error handling procedures exist, we need to schedule our network in such a way that it tolerates maximum number of errors. If we schedule considering error possibility, even if some bit errors happen in the bus, no message misses its deadline.

2.3 Performance Parameters

We obtained performance metrics from literature research that are important for the CAN protocol. We check whether our solutions satisfy these performance metrics or not.

-Guarantee for timing constraints: Message sets should be schedulable. Before mounting the message set to the in-vehicle network, response time analysis should be conducted elaborately and no messages should miss its deadline.

-WCRT analysis accuracy: Analysis should give correct results. This results could be exact or approximate. Sometimes upper bounds are accepted as approximate results. But these upper bounds are too pessimistic, they can give unschedulable results even if the system is schedulable.

-Algorithm efficiency: One of the most important parameters is algorithm efficiency. Complexity of algorithms cannot be high, because CAN networks cn be medium or large sized. For example, if the complexity is exponential, it takes hours to complete the analysis. In most of CAN algorithms, there is a trade-off between complexity and exactness. If we want exact results, complexity could be very high. However,if we want complexity to be small, then instead of exact results, we should be satisfied with bounds. Note that these bounds should be safe and tight.

-Number of errors tolerated: Network should be schedulable and also the difference between the message deadline and response time must be maximum. This difference is called as slack. If an error occurs, it can be tolerated due to slack. As the difference increases, so does the number of errors tolerated.

-Delay tolerated: Some delays are possible in practical applications such as buffering delay, copy time delay or delay due to error. If we have high message slacks, then delay can be tolerated. We should choose the delay tolerant message schedulings.For example, for messages A,B,C,D,E, with different orderings, we have different sum of slacks. Deadline minus jitter priority ordering algorithm produces A-B-C-D-E order. In that order sum of slacks are 74 time units. Robust priority ordering algorithm produces A-C-B-D-E order with sum of slacks 110 time units. Second ordering tolerates

a delay of 110 time units while the first one tolerates 74 [20].

-Bus utilization: Bandwidth should not be wasted, achievable bus utilization should be as maximum as possible. In CAN, in the past above 40 % was considered as high bus utilization, but today with efficient algorithms, 60-80 % is considered as high.

-Bus speed required: Minimum bus speed required means that what should be the bus speed if we want maximum utilization at minimum speed. Below this value, the system is not schedulable anymore. CAN bus speed and bus length are inversely proportional. With 40 meters cable, we can have at most 1 Mbps. If we want a bus of 500 meters long, then we can have at most 100 Kbps speed. Bus length is an important parameter for car manufacturers. CAN bus connects many parts inside a car and it may be needed to have a long bus. If we increase the length, then we need to reduce the speed. However, we cannot reduce below the minimum bus speed required.

-Runtime CPU overhead: If the algorithmic complexity is high, it takes some time to run the code. Even if CAN analysis and scheduling algorithms work offline, still they have to finish in a reasonable amount of time for practical use in industrial environment by network designer.

2.4 Literature Search

The timing analysis has been elaborately studied before. Bounds on the worst case response times were first specified in [41], then refuted in [19]. There were some faults in [41] and [19] fixed it. Also optimal priority assignment method were proposed in the same paper. But these methods are developed for ideal CAN. Later researches have integrated the practical limitations of hardware [25, 38, 18, 35] . CAN with FIFO queues are studied in [25, 38]. Finite buffer space problem is solved in [18]. Another practical problem is non-abortable transmission requests, this problem is discussed in [35]. Robustness issue is investigated in several papers [39, 23, 20] and robust scheduling and extension algorithms are proposed. For FIFO extension, [39] paper is taken as reference.

In [31], scheduling messages with offsets idea is proposed. There are several methods

in the literature that analyze these worst-case response times assuming that appropriate offsets are assigned to CAN messages [26, 47, 31, 17, 33]. Experimental evaluations are performed in [31] and approximations are given in [17, 26]. Exact offset analysis is studied in [47]. In addition, the case of FIFO queues instead of priority queues is discussed in [44]. In the scope of this thesis, the algorithms in [26, 47, 17] are implemented in order to support the schedulability analysis for CAN with offsets. [31] proposed offset scheduling algorithm and [16] further continued on this research and came up with offset scheduling solutions. Also we implemented [16] algorithms.

CAN-to-CAN(CAN2CAN) gateways are considered in [25, 46, 45]. An analytical approach for estimating the worst-case response time of CAN messages that pass gateways is proposed in [45]. Nevertheless, the existing literature does not give any scheduling algorithm. [45] is implemented in this thesis and for developing scheduling and FIFO analysis methods, we used [25] paper.

2.5 Computational Model and Notation

We consider messages that are transmitted on the CAN bus. Each message m is specified by a tuple $m = (C_m, T_m, J_m, D_m, E_m)$, whereby C_m is the maximum transmission time, T_m is the message period , J_m is the release jitter, D_m is the deadline and E_m is the transmission deadline of m. C_m is evaluated according to (2.1), T_m represents the time between two message generations for periodic messages and the minimum inter-arrival time for sporadic messages, J_m represents the maximum time from the generation of m until its availability in the CAN device driver queue, D_m states the maximum allowable time between the generation of m until its successful arrival at a receiver node and $E_m = D_m - J_m$.

Regarding scheduling, we consider a CAN network with a set of messages $\mathcal{M}$. Since schedulability on CAN depends on the relative priority order of messages instead of the absolute CAN ID [41, 19, 39], we introduce the priority order $o : \mathcal{M} \mapsto \{1, \ldots, |\mathcal{M}|\}$. Here, $o(m)$ represents the priority level of message $m \in \mathcal{M}$. The lowest priority level is $|\mathcal{M}|$ and the highest priority level is 1. For a message $m \in \mathcal{M}$, we write $\mathrm{lp}(m)$ for the set of lower-priority messages than m: $\mathrm{lp}(m) = \{m' \in \mathcal{M} | o(m') > o(m)\}$; $\mathrm{hp}(m)$ is

the set of higher-priority messages than m: $\text{hp}(m) = \{m' \in \mathcal{M} | o(m') < o(m)\}$.

Depending on the priority order o, each message $m \in \mathcal{M}$ has a WCRT denoted as R_m. We say that a priority order o is feasible for a message $m \in \mathcal{M}$ if $R_m \leq E_m$. If o is feasible for all $m \in \mathcal{M}$, o is called feasible. That is, the goal of message scheduling on CAN is determining a feasible priority order o.

The notation introduced in this section is summarized in Table 2.1.

Table2.1: Notation

Message under analysis denoted as m	
C_m	Longest transmission time
D_m	Deadline
J_m	Release jitter
E_m	Transmission deadline
T_m	Period or minimum inter-arrival time
w_m	Queuing delay
R_m	Worst-case response time
$hp(m)$	Higher-priority messages than m
$lp(m)$	Lower-priority messages than m
B_m	Blocking time
o	Priority order

CAN messages could be periodic or sporadic. In our analysis, we assume that sporadic messages are periodic with minimum inter-arrival time. This adds pessimisms to calculations.

2.6 Traditional Schedulability Analysis

There are 3 major duties of a CAN network designer:

-Packing signals into CAN messages.
-Calculating WCRT's of CAN messages and ensuring that these messages do not violate their deadlines.
-Assigning Id's to CAN messages such that these messages arrive to their destination nodes before violating the deadlines.

The first one is out of the scope of this thesis. We assume that signals are already packed to CAN messages. But the later two are studied in detail. In the thesis, calculating WCRT's of messages is called as CAN analysis or schedulability analysis and assigning Id's to CAN messages is called as CAN scheduling.

In this section, we try to compute the WCRT of messages on the CAN network. The computed values are used for checking the scheduability of the network by comparing these values with deadlines. If WCRT's of all messages are smaller than deadlines, the network is schedulable. Even if one of the messages violates the deadline, then the whole network is unschedulable.

Worst case response time *Rm* of any message *m* consists of 3 elements:

$$R_m = J_m + C_m + w_m \tag{2.2}$$

One of the elements is J_m, message may be released later than expected. w_m is the queueing delay.It is the time that the message has to wait until the start of transmission. The last element is maximum transmission time C_m. It depends on bus speed and message length.

Queuing delay is composed of two elements: blocking delay and interference delay. CAN protocol uses priority based arbitration, always higher priority messages start transmission before lower priority messages. But if a lower priority message is on the bus just before a high priority message, the higher priority message has to wait until the end of the transmission because of the non-preemptive nature of CAN. This waiting time is called blocking delay for that high priority message. In other words, blocking delay is due to one of the lower priority messages. In 2.3, it is seen that blocking message is the longest message among the lower priority messages.

$$B_m = \max_{k \epsilon lp(m)} (C_k) \tag{2.3}$$

Interference delay is because of higher priority messages which wins arbitration before message *m*. The calculation of interference delay is more complicated than blocking delay. It is not simply the addition of transmission time of higher priority messages, because more than one instance of same higher priority message may

be transmitted before m. Therefore, we have to calculate busy period. Busy period is defined as continuous interval of time during which no lower priority messages starts transmission and ends when the bus is idle. The initial value for priority level-m busy period is $t_m = C_m$ and it finishes when $t_m^{n+1} = t_m^n$.

$$t_m^{n+1} = B_m + \sum_{\forall k \epsilon hp(m)} \left\lceil \frac{t_m^n}{T_k} \right\rceil C_k \tag{2.4}$$

After calculating the busy period, we need to know how many instances of message m is released during busy period. Then we calculate response time for each instance and the largest of the response time values is the worst case response time of message m. Number of message instances in the busy period is given as:

$$Q_m^n = \left\lceil \frac{t_m}{T_k} \right\rceil \tag{2.5}$$

2.6 gives the queueing delay of q th instance. The equation starts with $w_m = B_m + qC_m$ and ends when $w_m^{n+1} = w_m^n$ or $w_m > D_m$.

$$w_m^{n+1}(q) = B_m + qC_m \sum_{\forall k \epsilon hp(m)} \left\lceil \frac{w_m^n}{T_k} \right\rceil C_k \tag{2.6}$$

WCRT is the largest of the response times of instances. Response time of q th instance:

$$R_m(q) = w_m(q) - qT_m + C_m \tag{2.7}$$

WCRT of message m is therefore:

$$R_m = \max_{q=0...Q_m} (R_m(q)) \tag{2.8}$$

2.7 Case Studies

Throughout this thesis, some algorithms and methods are developed, implemented and compared with existing algorithms and methods. In order to confirm accuracy, efficiency and effectiveness of the proposed methods and algorithms, we conduct some experiments. For these experiments, we need message sets to test our methods and algorithms. We also use an analytical tool and a simulator.

2.7.1 Netcarbench

This software generates example message sets which are similar to messages used in real cars. We have used the message sets generated by Netcarbench for the assesment of our algorithms ([8]). It is an open-source software and we have used its source code and embedded it to our own code. By giving the properties of the network such as bus speed and bus load, we can generate the message sets.

2.7.2 NetCarAnalyzer

Car manufacturers make tests before deploying CAN network to the vehicle. Since it is a real-time system, they have to be sure that messages meet their deadlines. For this purpose, they use analytical timing analysis tools. There are many commercial tools, one of the is NetCarAnalyzer ([7]). NetCarAnalyzer is a timing analysis tool that supports various forms of CAN networks such as CAN with or without offset, CAN with FIFO queues or priority queues. It uses the latest researches in the literature and among its customers there are many big companies such as PSA Peugeot-Citroën and Renault ([7]). For the verification of our implementations, we use this tool.

2.7.3 RtaWSimulator

RtaW simulator enables to simulate CAN networks ([9]). For the evaluation and verification of our implementations, in some cases we use this simulator. RTaW-Sim is able to simulate and predict the performances of CAN networks. These networks

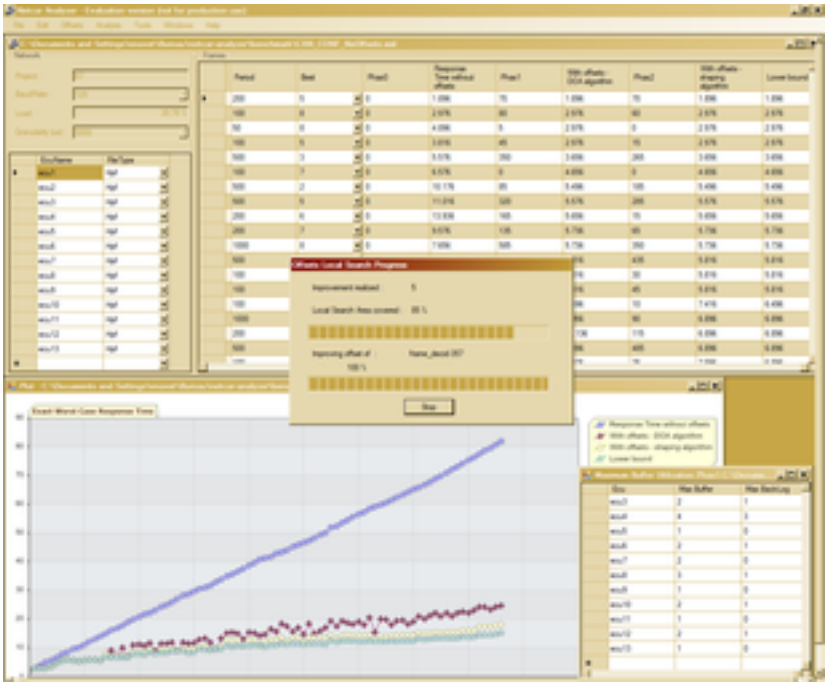

Figure 2.1: Screenshot of NetCarAnalyzer

could be interconnected via gateways or could generate errors. We use starter version of this simulator since it is free. The non-free version has more capabilities. RTaW-Sim helps the network designer compare different design solutions, choose the right components (e.g., waiting queue policy) and communication controllers (e.g., number of buffers), and configure them. But we only use for evaluating WCRT of our networks. Both NetCarAnalyzer and RTAW-Sim take NetcarBench output as input. NetCarBench produces .xml files suitable for both of them.

CHAPTER 3

CAN OFFSET ANALYSIS

3.1 Overview

This chapter reviews the offset concept for CAN networks and its analysis and scheduling methods. Offset definition and its effects are studied in Section 3.2. In the literature, there are several methods for worst case response time analysis for CAN systems with offsets. In Section 3.3, 3.4 and 3.5 three schedulability analysis methods are studied and implemented. One of these methods gives exact results and the other two give approximate results. The exact method has a high algorithmic complexity and is not practical and we consider the approximate methods as a more practical solution. Topic of assigning proper offsets is studied in Section 3.6. Three offset assignment algorithms are reviewed and adjusted according to results of the previous sections that we show. These offset assignment algorithms use schedulability analysis and using a better analysis improves results. Hence, we propose an algorithm that presents a contribution to CAN offset analysis in Section 3.7: we have assigned offsets and priorities together, which is a new topic and does not exist in the literature before us. Finally, Section 3.8 contains the case studies of this chapter. Three of these existing schedulability analysis methods are compared and offset assignment algorithms are evaluated according to these methods. Moreover, our contribution on priority and offset assignment is tested and compared.

3.2 Offset Definition

Offset is the relative time difference among the first generation of the messages. A message m_i is released at $t = 0$ and when another message m_j is released with a delay at $t = O_j$, this O_j is called offset. By using offsets, we try to spread the message load over time and reduce the maximum peak load. As argued by [31], offsets dramatically reduce the WCRT's of messages. Normally, while doing WCRT analysis, we assume that all higher priority messages are released synchronously. It is possible that for a certain period, bus becomes highly loaded, some messages miss their deadlines and after that period bus becomes idle. If we break the release synchronization by sending messages asynchronously, we can reduce WCRT. The bandwidth of the bus is then shared fairly by all messages. Then we can further increase the maximum bus load. Offset usage increases the bus load from 40% to up to 80% with a good offset scheduling algorithm [31]. Offset assignment to messages is called *offset schedule* .

The traditional schedulability analysis does not support offsets. There are several methods in the literature for WCRT analysis. In these methods, it is assumed that messages are already assigned offsets and priorities. We investigate three different WCRT algorithms for CAN systems with offsets and compare them. An important parameter here is the algorithm complexity. Although such kinds of algorithms work offline, still they have to end in a bounded time. Each of them has different algorithm complexity and exactness. There is a trade off between complexity and exactness. We choose one of them as the superior one. Also there are several offset assignment algorithms. Offset scheduling decides the offset for each message so as to make a given CAN message set schedulable.

In Figure 3.1, it is seen that messages are released with offsets. There are two nodes, m_1 and m_2 are released from one node and m_3 from the other. In the second part of the figure, m_1 has offset value 1 and m_3 has 2. When there were no offsets, WCRT of the message m_3 is 3, but when there is offsets, its WCRT decreases to 1. Similarly, WCRT of m_2 decreases from 2 to 1. This example simply shows the effect of offsets.

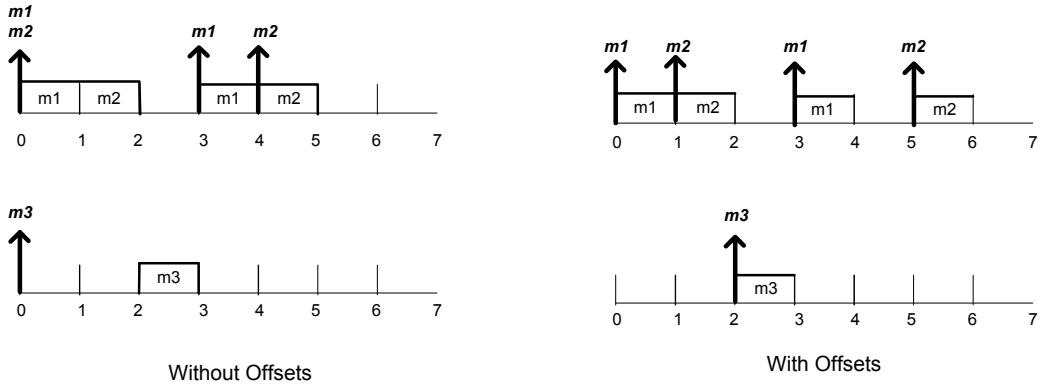

Figure 3.1: An example CAN Network with Offset Assigned Messages

3.3 Response Time Analysis with Offsets: Exact Method

This analysis method described here is based on [47]. We studied and implemented the analysis method in that paper in code. It gives exact WCRT's but its algorithmic complexity is high. It has exponential complexity. It is only suitable for small and medium sized networks.

In CAN protocol, each node has its own clock, there is no time synchronization among nodes. Hence, offset idea is only valid within the node. Even if their offsets are different, two messages from different nodes can be released simultaneously. We do not know the origin of timeline for each node and this makes the analysis complicated.

In order to find the WCRT of messages with offsets, the exact analysis uses *the transaction model*. Each node is modeled with a transaction. The least common multiple (LCM) of periods of messages of the same node is called a *hyperperiod*. If we consider message instances of that node during a hyperperiod, these instances constitute the transaction. Only one hyperperiod is enough because transactions are periodic with the hyperperiod. In Figure 3.2, a timeline of length 24 is shown. LCM of periods of messages is 12, and hence the hyperperiod is 12. The message pattern from $t = 0$ to $t = 11$ is an exact replica of the pattern from $t = 12$ to $t = 23$.

In the analysis, after transactions are formed, scenarios are decided. In a scenario,

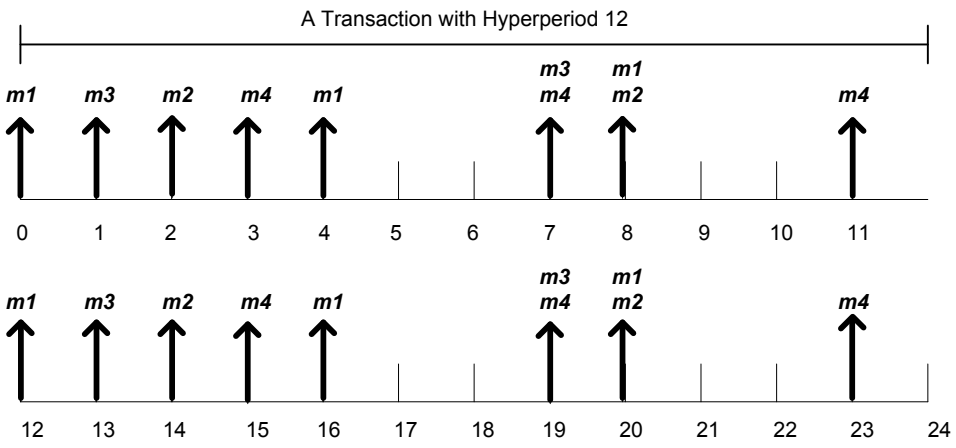

Figure 3.2: Transaction with Hyperperiod 12

each message instance from each transaction is a candidate for the time origin of that node. Response time can be calculated for each of these scenarios and the largest response time gives WCRT. But this leads to many possible scenarios and if the message set gets bigger, it becomes impossible to investigate all scenarios. Therefore, we need to reduce the number of scenarios. In the traditional analysis, we only consider the worst case and for the worst case, we assume that all higher priority messages are released simultaneously at a time called critical instant. That critical instant is too pessimistic for the offset case, because offset means asynchronism and messages are released at different times. Hence we redefine the critical instant. Critical instant is the instant at which at least one higher priority message is released from each node. We do not know which messages are released at the critical instant. Each higher priority message instance of each transaction is a candidate as critical instant message for any given message.

In short, in the exact analysis, we analyze all worst case scenarios and in each of these scenarios a different higher priority message is assumed to be released at the critical instant.

3.3.1 Response Time Of A Scenario:

We are analyzing a scenario for worst case response time of message m at priority level l. m is released at critical instant and candidates from other nodes are also released at that instant. Critical instant candidates are modeled with ϕ_i variable. ϕ_i is the property of the node. It is the time difference between the start of transaction and the critical instant. With each critical instant change, we have a different ϕ_i choice for that node.

Relative positioning of messages to the critical instant point is shown with φ_i. It is a property of message. For each scenario, each message of each node has different φ_i values. It can be calculated as follows:

$$\varphi_i^k = (T_i^k - (\phi_i - \phi_i^k))(mod\ T_i^k) \tag{3.1}$$

Table 3.1 shows an example message set and Figure 3.3 displays the transactions of nodes. Let see from Table 3.2 all possible scenarios for the given example together with possible ϕ_i and φ_i values :

Table3.1: Message Set With Two Nodes

Node 1			Node 2		
Message	Period	Offset	Message	Period	Offset
m_1	4	0	m_0	4	0
m_2	6	2	m_5	8	2
m_3	6	1			
m_4	8	3			

For all 6 messages, we have to analyze 24 scenarios. It means a total of 144 iterations. For scenario 1, time origin of each transaction is zero. Therefore, at the critical instant $t = 0$, m_1 and m_0 are released. At $t = 1$, m_3 is released. m_2 and m_5 are released at $t = 2$ and m_4 is released at $t = 3$. For scenario 2, critical instant is $t = 1$ for node 1 and $t = 0$ for node 2. m_3 and m_0 are released at the critical instant for this scenario. For each of these scenarios, we do not have to make computations. As we mentioned, we need only worst case scenarios. Only higher priority messages are considered. For

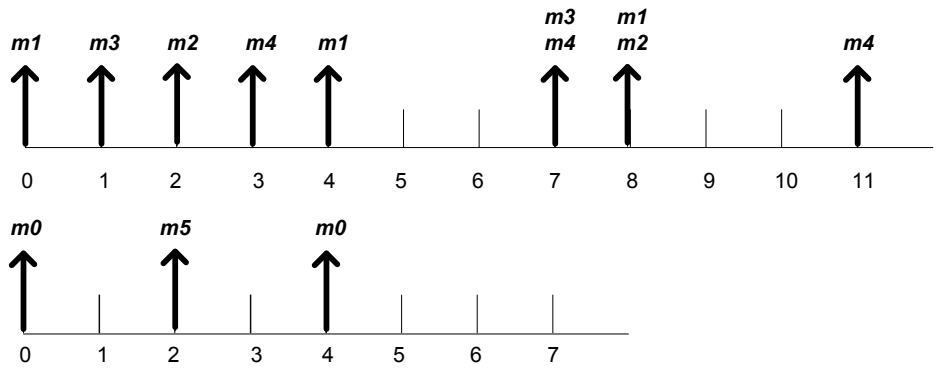

Figure 3.3: Transactions of 2 Nodes

example, for the possible scenarios for m_3, we are interested in only $\phi_1 = 0,1,2,4,7,8$ and $\phi_2 = 0,4$. Therefore, number of scenarios is reduced to 12 for message m_3.

For each worst case scenario, we calculate WCRT of the message under analysis. Response time is the sum of queuing delay and transmission time. Queuing delay of m consists of three parts: First one is blocking delay due to lower priority messages. Second one is interference due to higher priority messages that released from other nodes at critical instant or later before the busy period ends. And the last one is interference due to higher priority messages that are released from the same node during the busy period. First, as in the traditional analysis, we should calculate the busy period. (3.3) shows busy period calculations. It is an iterative equation, initial value starts with $t = C_m$ and ends when $t^n = t^{n+1}$. In this equation, second term is due to transmission time of the instances of the message under analysis and third term is due to interference due to higher priority message instances. $M(t)$ value gives the message instances during busy period and it is calculated with (3.2). Its value depends on the critical instant choice. For each higher priority message, we need $M(t)$ value. It is calculated during iterative computations.

$$M(t) = \left\lfloor \frac{J_i^k + \varphi_i^k(\phi_i))}{T_i^k} \right\rfloor + \left\lfloor \frac{t - \varphi_i^k(\phi_i))}{T_i^k} \right\rfloor + 1 \tag{3.2}$$

$$t = B_u^a + M_u^a(t)C_u^a + \sum_{\Gamma_i \epsilon \Gamma, F_i^k \epsilon hp_i(F_u^a)} M_i^k(t)C_i^k(t) \tag{3.3}$$

Table3.2: Possible Scenarios

Scenario No	ϕ_1, ϕ_2	φ_0	φ_1	φ_2	φ_3	φ_4	φ_5
1	$\phi_1 = 0, \phi_2 = 0$	0	0	2	1	3	2
2	$\phi_1 = 1, \phi_2 = 0$	0	3	1	0	2	2
3	$\phi_1 = 2, \phi_2 = 0$	0	2	0	5	1	2
4	$\phi_1 = 3, \phi_2 = 0$	0	1	5	4	0	2
5	$\phi_1 = 4, \phi_2 = 0$	0	0	4	3	3	2
6	$\phi_1 = 7, \phi_2 = 0$	0	1	1	0	0	2
7	$\phi_1 = 8, \phi_2 = 0$	0	0	0	5	3	2
8	$\phi_1 = 11, \phi_2 = 0$	0	1	2	1	0	2
9	$\phi_1 = 0, \phi_2 = 2$	2	0	2	1	3	0
10	$\phi_1 = 1, \phi_2 = 2$	2	3	1	0	2	0
11	$\phi_1 = 2, \phi_2 = 2$	2	2	0	5	1	0
12	$\phi_1 = 3, \phi_2 = 2$	2	1	5	4	0	0
13	$\phi_1 = 4, \phi_2 = 2$	2	0	4	3	3	0
14	$\phi_1 = 7, \phi_2 = 2$	2	1	1	0	0	0
15	$\phi_1 = 8, \phi_2 = 2$	2	0	0	5	3	0
16	$\phi_1 = 11, \phi_2 = 2$	2	1	2	1	0	0
17	$\phi_1 = 0, \phi_2 = 4$	0	0	2	1	3	4
18	$\phi_1 = 1, \phi_2 = 4$	0	3	1	0	2	4
19	$\phi_1 = 2, \phi_2 = 4$	0	2	0	5	1	4
20	$\phi_1 = 3, \phi_2 = 4$	0	1	5	4	0	4
21	$\phi_1 = 4, \phi_2 = 4$	0	0	4	3	3	4
22	$\phi_1 = 7, \phi_2 = 4$	0	1	1	0	0	4
23	$\phi_1 = 8, \phi_2 = 4$	0	0	0	5	3	4
24	$\phi_1 = 11, \phi_2 = 4$	0	1	2	1	0	4

L_p is the busy period and from busy period and starting time, number of instances is found as:

$$\varepsilon^a_{u,L_p} = \left\lceil \frac{L_p - \varphi^a_u(\phi_i))}{T^a_u} \right\rceil \tag{3.4}$$

Due to jitter, the lowest numbered instance may have negative value. It is:

$$\varepsilon^a_{u,0} = -\left\lceil \frac{J^a_u - \varphi^a_u(\phi_i))}{T^a_u} \right\rceil + 1 \tag{3.5}$$

Then we calculate the start time of nth instance:

$$s^{a,n}_u = B^a_u + (n - \varepsilon^a_{u,0})C^a_u + \sum_{\Gamma_i \epsilon \Gamma} M^k_i(t) C^k_i(t) \tag{3.6}$$

For each instance,we find the response time using equation:

$$R_u^{a,n} = s_u^{a,n} + C_u^a(\varphi_u^a(\phi_i) + (n-1)T_u^a) \tag{3.7}$$

The largest response time gives WCRT:

$$R_u^{a,n} = \max\left\{R_u^{a,n} : n = \varepsilon_{u,0}^a, ..., \varepsilon_{u,L_p}^a\right\} \tag{3.8}$$

3.4 Response Time Analysis with Offsets: Approximate Method

The exact algorithm considers all possible worst case scenarios and it has exponential complexity. Table 3.3 shows number of worst case scenarios for two different example message sets. Although the number of worst case scenarios are extremely large, these sets are smaller than the real applications. Moreover, increasing the number of messages causes an exponential increase in the number of scenarios. Therefore, it is not suitable for larger networks. There is another solution for offset WCRT analysis.

Table3.3: Two example message sets

Message Set	Number of Scenarios
3 Nodes-24 Messages	171.800
5 Nodes-45 Messages	90.902.150

Instead of working with all possible worst case scenarios, an approximation method can be employed. If this approximation is safe and run-time is bounded, then this method is a good solution. Safety means that the approximate WCRT is always larger than the exact WCRT. Moreover, approximate results should be tightly bounded, there should not be too much difference between the approximate WCRT and the exact WCRT. Because even if the system is schedulable, the approximate method may give unschedulable results.

WCRT of a message consists of contributions from other nodes. Instead of taking contributions of each *message* to the WCRT separately as in the exact case, this method

uses contributions of *nodes*, which reduces the complexity dramatically.

The message under analysis m is released and we add contributions of other nodes to the WCRT of m. Contribution from a node can be in one of two forms: blocking message and synchronized message. A lower priority message from other nodes may win arbitration just before the release of m. This is the blocking message. Higher priority messages from other nodes that are released simultaneously with m are called synchronized messages.

In the nodes, initially blocking and synchronized messages are unknown and they are found by trial and error. At the beginning, we choose some blocking and synchronized messages and find their contributions. With these initial contributions, we calculate an initial response time. Then process continues with iteration. Given the initial value of response time and contribution of each node to this value, we alter the synchronized message in a node if this change increases the response time and the contribution of this node. All higher priority messages from other nodes are candidates for synchronized message. This trial and error method is called worst case exploration algorithm (WCEA).

$$C_{max}(i) = \max_{k \epsilon hps(m,i)} (C_k) \tag{3.9}$$

$$B_{max}(i) = \max_{k \epsilon lps(m,i)} (C_k) \tag{3.10}$$

For each node, among higher priority messages, we find C_{max}. Similarly, for each node among lower priority messages, we find B_{max}. Then blocking delay is found as:

$$B_m = \max_{1 \leq i \leq N_{max}, i \neq N_m} (B_{max}(i) - C_{max}(i)) \tag{3.11}$$

While calculating blocking delay, we subtract C_{max} because later, we add it for calculating queuing delay. Initial contributions are found according to the formula:

$$W_m^0 = C + B_m + \sum_{1 \le i \le N_{max}, i \neq N_m} C_{max}(i) \tag{3.12}$$

Our initial assumption is that contributions of nodes are due to longest messages. However, the worst case may not appear when the longest messages are released together. We need to try other messages and check whether the contribution of that node increases if we change the message. We continue the iteration until $w_m^n = w_m^{n+1}$ or $w_m > D_m$, in the later case, the system is unschedulable.

$$W_m^k = W_m^0 + \sum_{1 \le i \le N_{max}} Ctrb_i^k \tag{3.13}$$

This method is proposed in [26]. The pseudo-code given in that paper is shown in Algorithm 1. We have implemented it for the case studies. The complexity of this algorithm depends on the number of messages, unlike exact analysis, it has linear complexity.

input : Message Set M with its features

output: WCRT of each message

for *each message m,* $m = 1$ *to* $m = |M|$ **do**
  for *each node i* **do**
    Find $B_{max}(i)$ and $C_{max}(i)$
  end
  Calculate B_m, w_m^0 and $Ctrb_i^0$
  while w_m^n *!=* w_m^0 **do**
    for *each node i* **do**
      for *each instance j* **do**
        Compute $Ctrb_j$ for instance j
      end
      $Ctrb_i = \max_{1 \leq i \leq N_{max}, i \neq N_m}(Ctrb_j)$
    end
    $w_m^n = w_m^0 + Ctrb_i$
  end
  $R_m = w_m + C_m$
  return R_m
end

Algorithm 1: Approximate Offset WCRT Analysis Algorithm

3.5 Response Time Analysis with Offsets: Maximum Interference Method

The last offset analysis method we study is the maximum interference function(MIF) analysis. The analysis described in this section is based on [17]. Interference function (IF) and MIF are mathematical representations that are used to formulate the WCRT calculations. Recall from the previous sections that we have used the transaction model for message instances of nodes and generated scenarios. There are many possibilities for the time origin of a transaction and for each node, each possibility is a scenario. Here, for each scenario we represent each node with IF's. It is a function of time and represents transmission times of message instances inside a transaction. It is like taking a screenshot of transactions for each scenario. In a time interval, if there is a message transmission, then the slope of the function is 1 and if there is no

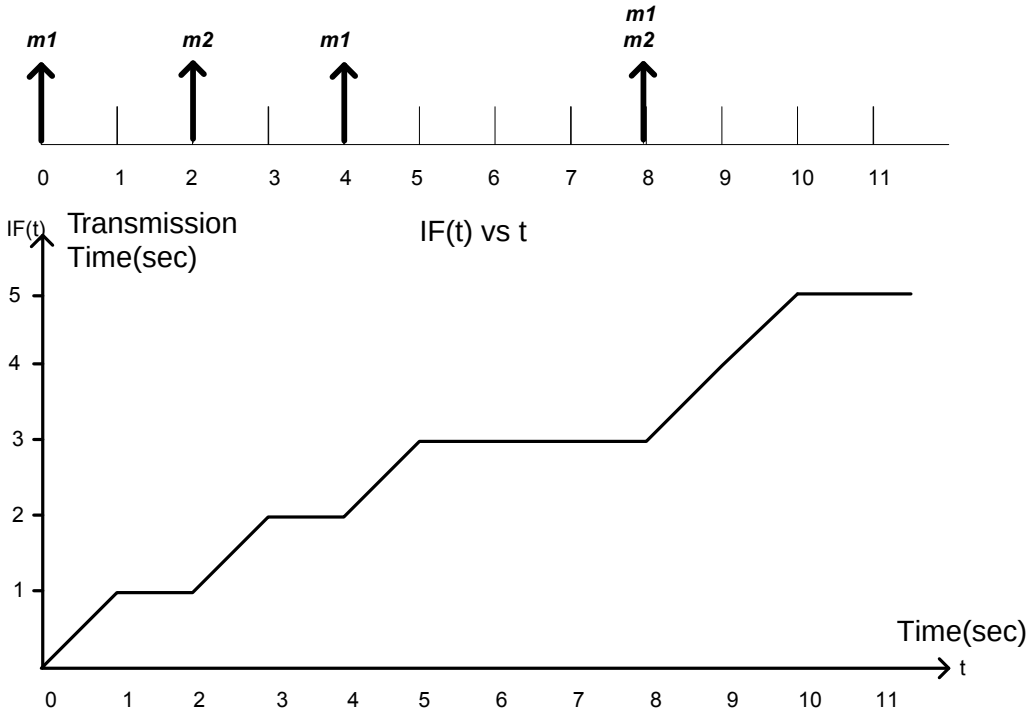

Figure 3.4: Obtaining IF From Transaction

transmission, the slope is 0. Figure 3.4 shows how to obtain IF from a transaction.

Note that in the IF, we only have the higher priority message instances. Assume that we want to calculate response time for a scenario (but not the worst case for simplicity). We first obtain the IF's of all nodes, then we can calculate response time by adding the IF's of all nodes. The addition is not a normal addition, it is called *saturation addition*. It is the operation of picking up the uppermost line of all IF lines. It provides adding up the delays of higher priority messages.

If we obtain IF of each node for each scenario and make the analysis for all possible scenarios and obtain the largest one as WCRT, then we are doing exact analysis. However, in the analysis, we use MIF instead of IF for a faster algorithm. By using MIF, we obtain a faster algorithm but approximate results. MIF is the saturation addition of IF's for each node. At the end of saturation addition, the message can win the arbitration when the slope becomes zero. Figure 3.5 shows how to obtain MIF from two given IF's. Addition starts from zero. Between $t = 0$ and $t = 1$, both of the IF's have slope 1, so MIF has slope between $t = 0$ and $t = 2$. Between $t = 1$ and

t = 1, IF2 has slope but IF1 has no slope. Hence MIF has slope between t = 2 and t = 3. Between t = 2 and t = 3, IF1 has slope, IF2 has no slope. Then, MIF has slope between t = 3 and t = 4. Between t = 2 and t = 3, both IF's has no slope, but MIF is already at t = 4. This process continues until MIF has no slope, at that point busy period ends and a lower priority message can gain access to the bus. We implement MIF algorithm using the pseudo-code given in Algorithm 2.

input : Message Set M with its features
output: Message WCRT's

```
for each message m, highest priority first do
    Obtain worst case offset possibilities
    for each offset possibility do
        Obtain the IF for the node of the analyzed message m
        Obtain the MIF's for all other nodes one by one
        Make saturation addition of all MIF's and IF's
        Add B_m and C_m values to saturation addition
        From the saturation addition, find the point where the slope becomes
        zero and it is equal to response time
    end
    Highest of the response times gives WCRT
end
```

Algorithm 2: MIF Analysis Algorithm

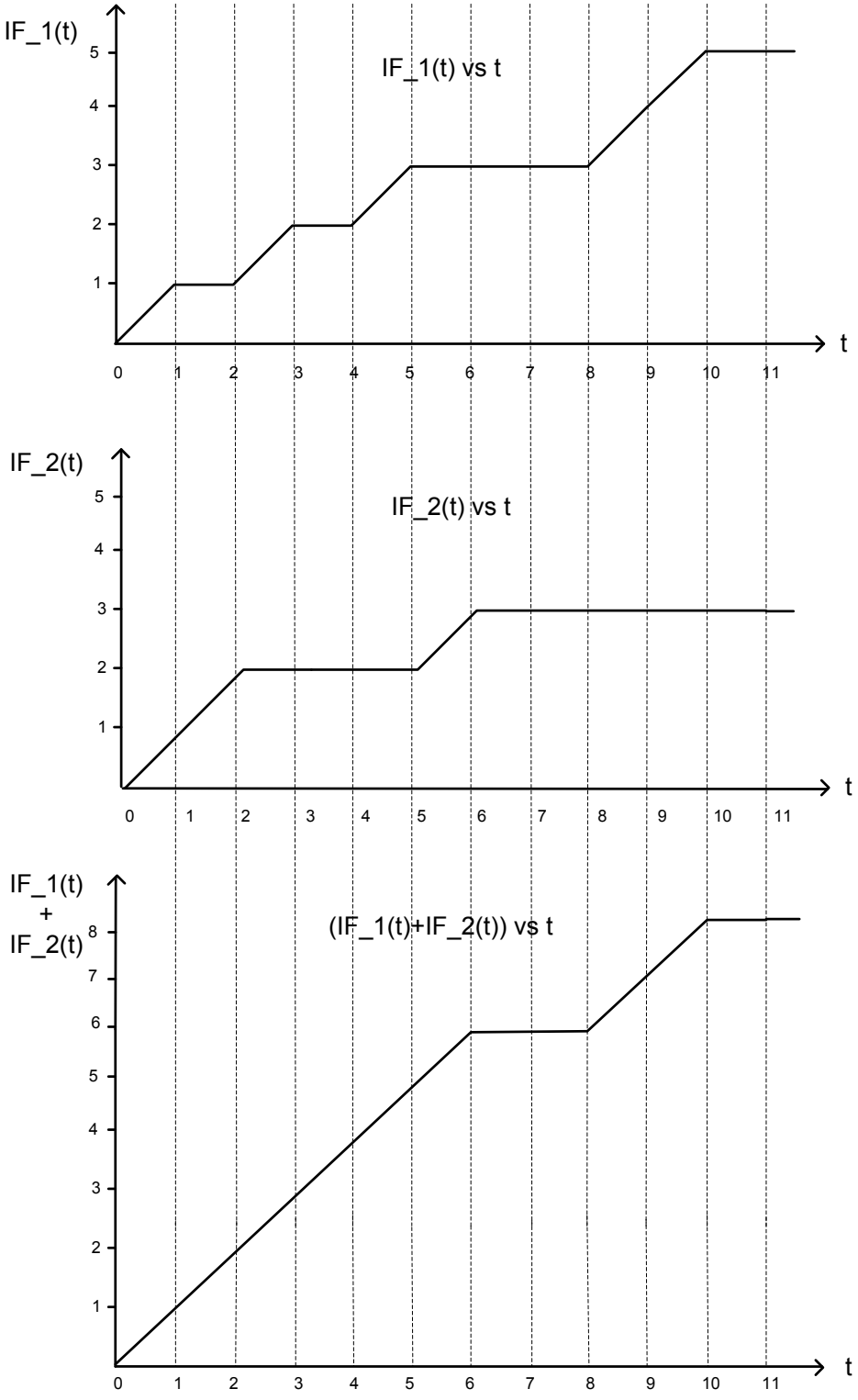

Figure 3.5: Saturation Addition of two IF's

3.6 Offset Assignment

Although offset usage increases the maximum bus load remarkably, using an inefficient scheduling algorithm may decrease this potential. With offset scheduling, we try to distribute load over time. Since nodes are not synchronized, distributing messages of each node uniformly does not mean a schedulable solution, even if it is schedulable it may not be a good solution. For example, several high priority messages with offsets from different nodes may meet at a certain time zone and some of them may violate their deadlines. Offset assignment is very similar to traffic shaping, it regulates the bus load to assure schedulability and increase total capacity. It provides a means to control the volume of traffic that is sent to the CAN bus. In this section, for offset scheduling we have reviewed three methods: two of them are proposed in [16] and one of them is from [31]. These three algorithms are: Standard Offset Assignment Algorithm (SOSA), Local Neighborhood Search Algorithm (LNSA) and Genetic Algorithm for Offset Scheduling (GAOS).

SOSA has the least complexity. It tries to spread the load as much as possible, but it ignores the effects of other nodes when assigning offsets. Therefore, other algorithms are proposed. LNSA assigns offsets by searching all possible offset combinations for all messages. For each combination an offset WCRT analysis is required, hence its run-time is high. GAOS uses the concepts of genetic programming. It is a machine learning algorithm and used to optimize the offsets.

3.6.1 SOSA

SOSA assigns offsets to each node separately. For each node, messages are put to timeline as far as possible from each other. This algorithm is proposed in [31]. For each message insertion, first the longest interval with the smallest load is decided for the corresponding node and new message is added to the middle of that interval. It works as follows: Assume T_{max} is the longest period in the analyzed node. First message is put to t=0 and the longest least loaded interval is $[0, T_{max})$, hence second message is put to $t = T_{max}/2$. For the third message, there are two options $T_{max}/4$ and $3T_{max}/4$. Put it to $T_{max}/4$ and continue until all the messages are placed.

3.6.2 LNSA

LNSA searches for offsets from a set of offset configurations. Time is divided into discrete intervals called granularity. Granularity 2 means, offsets are multiples of 2. Each message can have offsets at multiples of granularity value. For example, consider a message set of 10 messages with granularity 2 and period 10. There are $(10/2)^{10}$ different combinations. For the best result, for all combinations, WCRT analysis can be done and the best one can be chosen, but testing WCRT for all combinations is impossible. LNSA also tests possible offset combinations but not all of them. It starts with SOSA algorithm. First each message is assigned an offset according to SOSA. Then for each message starting from the highest priority, all offset possibilities are tried assuming that all other messages keep their SOSA assigned offsets. For each of these offset possibilities, WCRT analysis is performed and the sum of slacks of all messages is calculated. Then the offset which produces the largest sum of slacks is chosen. This offset is kept as the offset of that message. The next message is selected and the same steps are repeated. The run-time of LNSA algorithm depends on the granularity and the number of messages. LNSA was proposed in [31] and its pseudo-code is given in Algorithm 3.

3.6.3 GAOS

GAOS finds an optimization for offsets by using genetic programming.We first summarize the basics of genetic programming related to our research: It is inspired by evolution theory. It starts with a set of individuals called population. Individuals are chosen from the population to create new populations. Newer ones are better than the elders because individuals are chosen according to their fitness, only the fittest individuals join to reproduction. There are main operators used in genetic programming: selection, crossover and mutation. Selection is the choice of parents according to fitness. Crossover is the combination of properties of parents during reproduction. Mutation is the random change of properties of individuals.

If we are to apply this genetic idea to offset assignment, we apply the following procedure: Assigning offset to each message is a combination and each combination

input : Message Set M

output: Message Offsets

Apply SOSA and obtain an initial offset configuration
Sort messages in increasing deadline order
for *each message m in sorted order* **do**
    $SumOfSlacks = 0$ $MaxSlackValue = 0$
    for *each possible offset o* **do**
        Make WCRT analysis **for** *each message m* **do**
            $SumOfSlacks += Slack_m$
        end
    end
    if $SumOfSlacks > MaxSlackValue$ **then**
        $O(m) = o$ $MaxSlackValue = SumOfSlacks$
    end
end

Algorithm 3: LNSA Algorithm

is an individual. We start with an initial population, we make WCRT analysis for the individuals of this population and find the fittests. The fittest individuals are the ones with largest sum of slacks. Then by applying crossovers and mutations to the fittest combinations, we obtain child combinations. Note that the sum of slacks of new individuals are larger than those of previous ones, because algorithm iterates towards better individuals. The algorithm continues until we find a schedulable solution or maximum number of iterations is reached.

GAOS has less number of iterations than LNSA, because in LNSA we search all possible directions. However, in GAOS as we iterate, in the next iteration, we know in which direction we continue to search.

We have compared three algorithms in terms of efficiency and complexity. SOSA does not make offset WCRT analysis, but the other two use WCRT analysis inside the algorithms. The original LNSA and GAOS use the approximate WCRT analysis algorithm described in the third section. We have implemented these algorithms also with MIF and exact analysis algorithms.

```
input  : Message Set Z, maxNoOfIterations, mutationRatio,
         crossoverRatio,initialPopulationNo,PopulationNo
output: Message Offsets
for i=1 to i=initialPopulationNo do
    Generate Initial Populations using mutations
end
for each individuals in the population do
    Make WCRT analysis
    if Schedulable Individual Exists then
        return schedulable
    end
    else
        Calculate sum of slacks
    end
end
for i=1 to i=maxNoOfIterations do
    Select two individuals with the largest sum of slacks
    Crossover the parents with crossoverRatio
    Apply mutations to children
    for each individuals in the population do
        Make WCRT analysis
        if Schedulable Individual Exists then
            return schedulable
        end
        else
            Calculate sum of slacks and choose the population with the largest slack individuals, total
            population cannot exceed PopulationNo
            if population >PopulationNo then
                Discard the ones with smallest sum of slacks
            end
        end
    end
end
```

Algorithm 4: GAOS Algorithm

3.7 Offset and Priority Assignment Together

Many studies have been carried out in priority assignment topic. However, there exists no priority assignment algorithm for the offset case. Offset scheduling increases the bus load and we can further increase that load if we can freely assign priorities. Priority and offset are two parameters that affect WCRT, and they are not independent.

If we have a message set with unassigned priorities and unassigned offsets, what we normally do is first assigning priorities to messages and then assigning offsets or vice versa. The two processes work independently and as a result, we may obtain poor results. For example, for a given message set, assume first we assign offsets using SOSA and then assign priorities in deadline monotonic order, but the system may be unschedulable. Or, we assign priority first and offset second. A schedulable system may not be obtained with this configuration because even if we assign the smallest deadline message to high priority, if all higher priority messages gather at the same offset, it still misses its deadline. Therefore, priority and offset should be handled together.

We have developed an algorithm to solve this problem. We apply SOSA algorithm first, and in each node, messages are uniformly distributed. Then starting from the lowest priority level, messages are tried one by one for the priority levels assuming all other unassigned messages are at higher priorities. For this testing, we assume that messages in the same node have offsets and messages from other nodes are released simultaneously with the message under analysis since the nodes are not synchronized. We find a schedulable message for each priority level if possible. If a schedulable message does not exist for the given priority level then the one with largest slack is chosen meaning that the one which violates the deadline the least is chosen. After all messages are assigned to priority levels, the system is tested with new offsets and priorities. If schedulable, then algorithm returns. If still it is not schedulable, then we need to modify the offsets. We determine the unschedulable messages and apply a method which is similar to GAOS. Within the limited number of iterations, we change the offsets of unschedulable messages according to GAOS.

With the algorithm whose pseudo-code is given in 5, we do not promise an optimal

input : Message Set M=all messages, Message Set H = 0 and Message Set $L = M$

output: Message Offsets and Message Priorities

```
1  Apply SOSA algorithm and obtain messages with new offsets O
2  for each priority level l, lowest first do
3      for each message m do
4          Calculate WCRT of message m assuming m is at priority level l and all other unassigned
           messages are at higher priorities. Also assume same node messages have offsets and other
           messages are released simultaneously.
5          if m is schedulable then
6              H = HUm, L = L/m, Priority(m)=l
7          end
8          else
9              Choose any m with smallest deadline violation. H = HUm, L = L/m, Priority(m)=l
10             schedulable=false
11         end
12         while schedulable do
13             Test Set M with Priority() and O()
14             if M is schedulable then
15                 return true
16             end
17             else
18                 Determine unschedulable messages
19                 for i=1 to i=maxNoOfIterations do
20                     Select two individuals with the largest sum of slacks, apply crossovers and
                       mutations
21                     for each individuals in the population do
22                         Make WCRT analysis
23                         if Schedulable Individual Exists then
24                             return schedulable
25                         end
26                         else
27                             Calculate sum of slacks and choose the population with the largest
                               slack individuals, total population cannot exceed PopulationNo
28                         end
29                     end
30                 end
31             end
32         end
33     end
34 end
```

Algorithm 5: Offset and Priority Assignment Algorithm

scheduling algorithm, but filling the gap of literature on priority assignment for CAN with offsets.

3.8 Case Study

3.8.1 Case Study 1

In order to compare exact algorithm, approximate algorithm and MIF algorithm in terms of run-times and exactness, we have made an experiment. In this experiment, we have used 20 message sets, bus loads are between 30% and 40%. Number of ECUs are between 4 and 7. Message periods are among $10, 20, 50, 100, 200ms$. Offsets are chosen such that messages of each node are uniformly distributed. The experiment is run for 20 message sets serially and ended in 50 minutes. Table 3.4 shows the run-time and exactness comparisons. According to it, exact algorithm run-time is incomparable with the other two. For larger message sets, the exact WCRT calculation becomes impractical. MIF and approximate algorithms have no problem with run-time. But in terms of exactness, MIF algorithm is much better. In order to compare the exactness, we have summed the WCRT's of messages for each algorithm. The MIF sum is 98.2% higher than the exact sum. Similarly, the approximate sum is 72.9% higher than the exact sum.

Table3.4: Exact, MIF and Approximate Algorithms in terms of run-times and exactness

	Exact	MIF	Approximate
Run-times	48.5 min	23 sec	17 sec
Similarity to Exact Case	100%	98.2%	72.9%

There are four parameters we have observed that affect run-times of algorithms. Number of messages, period, number of ECU's and granularity. The number of total messages is the first parameter for the run-time. We have two message set groups. Each group has 10 different message sets. Each message set has same number of ECU's and same granularity, the number of nodes is 4 and granularity is 20. The bu speed is 125 Kbps. Both groups have standard periods as 10, 20, 50, 100, 200 ms. Average number of messages in the message sets of the first group is between 28 ad

32. Average number in the second group is between 43 and 46. Two message set groups have significant run-time difference, especially for the exact algorithm. Table 3.5 shows the comparison of two message set groups. Table shows the average of all message sets. As can be seen, as the number of total messages increases, run-times also increase. After increasing the number of messages by 50%, the run-time of the exact algorithm increases by 6 times. However, MIF and approximate run-times are still much smaller.

Table3.5: Run-time Comparison of Three Analysis Algorithms For Different Number of Messages

	Exact Analysis Average	MIF Analysis Average	Approximate Analysis Average
Group 1	49.4 sec	2.62 sec	0.358 sec
Group 2	277 sec	5.11 sec	0.608 sec

The number of ECU's is another important parameter for the run-time. We have compared two message set groups with nearly same number of messages. Again each group has 10 message sets, the number of messages in the message sets is between 28 and 32. The first group message sets have 3 ECUs and the second group sets have 6. Periods and granularity are same for both groups, granularity is chosen as 20. As can be seen from Table 3.6, as the number of ECUs increases with same number of total messages, run-times also increase because the number of scenarios increases which depends on the formula given by (3.14).

$$\prod_{j=NumberOfNodes} \left(\sum_{i=NumberOfMessages} \frac{Hyperperiod[j]}{PeriodOfMessage[i]} \right)! \tag{3.14}$$

Table3.6: Run-time Comparison of Three Analysis Algorithms For Different Number of ECU's

	Exact Analysis Average	MIF Analysis Average	Approximate Analysis Average
Group 1	9.67 sec	1.87 sec	0.25 sec
Group 2	614.35 sec	3.4 sec	0.28 sec

Another important parameter we have observed that affects run-times of the algorithms is the offset granularity. For a message set group of 10 message sets, we have

run the three algorithms for granularity of 10 and for granularity of 1. Group 1 has granularity of 10 and Group 2 has granularity of 1. The number of messages in each set is between 28 and 32, periods are again standard periods. Number of ECU's is 4. In Table 3.7, run-times are compared. As the granularity increases the offset possibilities decreases and run-times decrease, as well.

Table3.7: Run-time Comparison of Three Analysis Algorithms For Granularity Values

	Exact Analysis Average	MIF Analysis Average	Approximate Analysis Average
Group 1	90.7 sec	2.88 sec	0.27 sec
Group 2	498.5 sec	3.9 sec	0.343 sec

The last important parameter that affects run-time is period. First message set group has standard periods as 10, 20, 50, 100, 200 ms and second group has non-standard periods as 15, 35, 60, 130, 210 ms. The number of messages in each set is between 20 and 22. The number of ECU's in each set is 3 and granularity is 10. As can be seen from Table 3.8, non-standard periods increase run-time significantly.

Table3.8: Run-time Comparison of Three Analysis Algorithms For Different Period Values

	Exact Analysis Average	MIF Analysis Average	Approximate Analysis Average
Group 1	4.92 sec	1.24 sec	0.187 sec
Group 2	779 sec	5.02 sec	0.234 sec

3.8.2 Case Study 2

In order to observe the effect of granularity on schedulability, we have made this experiment. High granularity is better in terms of run-time. However, as the granularity increases, the number of unschedulable messages may increase. In Table 3.9, we have three granularity options. For schedulability comparison, we have chosen bus load between 70% and 90%. The first row has granularity of 10, second row has granularity of 5 and third row has 1. We have 5 message set groups and each group has 20 message sets. The number of messages is between 100 and 125 in each set. For WCRT analysis, we have used MIF offset analysis algorithm. In the table, the

number of unschedulable messages in each message set groups is shown. As can be seen from table, there is almost a direct correlation between granularity and number of unschedulable messages.

Table3.9: The Effect of Granularity On the Schedulability

	Number of Unch. Mess.				
Number Of Mess. In Each Set	100-105	105-110	110-115	115-120	120-125
Granularity=10	0	4	8	12	19
Granularity=5	0	2	7	9	13
Granularity=1	0	0	3	6	11

3.8.3 Case Study 3

We have made an experiment in order to assess the efficiency and complexity of offset assignment methods, SOSA, LNSA and GAOS. For this purpose, we have generated 6 different message set groups. These groups have different features as can be seen from Table 3.10. Using these sets, we have evaluated the three algorithms in terms of schedulability, number of schedulable messages and run-times. Each message set group has 20 message sets. Also for this experiment for evaluating schedulability, we have used MIF analysis method. This due to the timing and exactness concerns.

Table3.10: Message Set Groups

Group 1	Nonstandard Periods, Infrequent Offsets, Normal Load
Group 2	Standard Periods, Frequent Offsets, Normal Load
Group 3	Standard Periods, Infrequent Offsets, High Load
Group 4	Standard Periods, Infrequent Offsets, Normal Load
Group 5	Nonstandard Periods, Frequent Offsets, Normal Load
Group 6	Nonstandard Periods, Frequent Offsets, High Load

The nonstandard periods are $15, 35, 60, 130, 210$ and standard periods are $10, 20, 50, 100, 200$. Frequent offsets have granularity of 1 and infrequent offsets have granularity of 5. Normal load is between 30% and 45%, high load is between 60% and 80%. Table 3.11 shows the average number of unschedulable messages and total sum of WCRT's of each message for each group and for each algorithm. For all groups, we have used

deadline monotonic order for priority assignment. CAN bus speeds are adjusted to 125 Kbps for simplicity.

Table3.11: Average Number of Unschedulable Messages and Sum of WCRT's For Each Group and Each Algorithm

Group No	SOSA		LNSA		GAOS	
	Sum Of WCRT's	No Of Unsch. Mes.	Sum Of WCRT's	No Of Unsch.Mes.	Sum Of WCRT's	No Of Unsch. Mess
Group 1	7082	20	7054	15	7197	7
Group 2	1820	0	1806	0	1924	0
Group 3	5429	46	5018	0	7124	8
Group 4	1725	0	1689	0	1877	0
Group 5	6211	16	5930	8	5839	4
Group 6	25820	180	24095	165	21470	115

In Table 3.12, it is seen that each algorithm has different run-time performance for each group. GAOS finds just a schedulable offset configuration, but LNSA searches the best configuration in terms of schedulability. SOSA has no concern on schedulability, it just spreads messages uniformly. This is the reason why LNSA has the highest run-times.

For Group 1, GAOS is the best choice although sum of slacks is better in LNSA. SOSA has more unschedulable messages and less slacks. This result is consistent with Vakkas study. In Vakkas, for Class A type message sets GAOS is suggested. For Group 2, SOSA seems ideal. Although sum of WCRT's are very close, run-times are comparably different. For Group 3, we can say that GAOS is good for such kinds of message sets, but if the run-time is not important LNSA can be used. If we compare with Vakkas study, for Class E type message sets, results are again consistent. For Group 4, we see that for such message sets run-time seems trivial. Hence LNSA is better.

If we compare Group 2 and Group 4, we see that increasing granularity decreases the run-time. The disadvantage is on schedulability. High granularity message sets are less schedulable than the sets with lower priority. By looking at Group 2 and Group 5 together, we observe that period is another factor that affects run-time. For Group 5 and Group 6, nonstandard periods deteriorate the system performance. If we compare Group 1 and Group 5, it is seen that frequent offsets and infrequent offsets affect the schedulability and frequent offsets are better in terms of schedulability.

Table3.12: Total Run-Time Values For Each Group and Each Algorithm

Group No	SOSA	LNSA	GAOS
Group 1	7.5 sec	280.2 sec	18 sec
Group 2	4.3 sec	61.4 sec	5.57 sec
Group 3	7.5 sec	25.8 sec	9.47 sec
Group 4	1.8 sec	4.4 sec	2.4 sec
Group 5	5.2 sec	3120 sec	23.6 sec
Group 6	39.2 sec	59576 sec	217 sec

3.8.4 Case Study 4

In Case Study 1 and Case Study 3, we have measured the performance and run-times of offset assignment and offset WCRT analysis algorithms. LNSA and GAOS algorithms use WCRT analysis methods inside their own algorithm and we need to choose the best analysis algorithms for both of them. In this experiment, we ran LNSA and GAOS algorithms together with MIF and approximate analysis algorithms and compared the performance metrics. We have also tried exact algorithm for both of them and decided that using exact analysis algorithm inside offset assignment algorithms is not practical. Because a single offset assignment algorithm runs WCRT analysis many times and in the exact case, the process requires several days.

Table3.13: LNSA and GAOS with both MIF and Approximate Analysis

Analysis Algorithm	LNSA		GAOS	
	Run-Time	No Of Unsch. Mes.	Run-Time	No Of Unsch.Mes.
MIF Algorithm	1963 sec	0	175.6 sec	2
Approximate Algorithm	126.6 sec	27	14.74 sec	73

In Table 3.13, a message set group of 20 message sets is investigated and average run-time for each set and total number of unschedulable messages are shown. The bus loads are between 70% and 90%. When we compare the rows of the table, it is clear that there is a trade-off between run-time and unschedulable messages. If we have no timing concern, then MIF is better. Also, we see that for this configuration LNSA is superior for both MIF and approximate cases.

3.8.5 Case Study 5

Using the algorithm developed in Section 1.7 for offset and priority assignment, we can achieve very high loads (above 90%) as long as message deadlines are equal to message periods. When we look at the message set groups given in Case Study 3, we see that Groups 1, 3 and 6 violate schedulability. We have applied our offset and priority assignment algorithm to these groups and found schedulable configurations for each of the message sets at 125 Kbps speed. Then, we have halved the message deadlines and again applied our algorithm. For each message set, we have found the minimum bus speed required in order to make the system schedulable using offset and priority assignment algorithm. At the found speeds, we have applied LNSA algorithm to the message sets for offset assignment. For the priority assignment, we have used deadline monotonic ordering. For those configurations, we have found the number of unschedulable messages. We have repeated the experiment, for $Deadline = Period/3$ case. In Table 3.14, for the found bus speeds, the number of unschedulable messages can be seen. Note that these are the result of LNSA algorithm with deadline monotonic ordering. For offset and priority assignment algorithm the number of unschedulable messages is all zero.

Table3.14: Minimum Bus Speed Required For Offset and Priority Together Algorithm and Number of Unschedulable Messages For LNSA Algorithm at These Speeds

GroupNo	Deadline=Period			Deadline=Period/2			Deadline=Period/3		
	Min. Bus Speed	Unsch. Mes. With LNSA	Unsch. Mes. With our Alg.	Min. Bus Speed	Unsch. Mes. With LNSA	Unsch. Mes. With our Alg.	Min. Bus Speed	Unsch. Mes. With LNSA	Unsch. Mes. With our Alg.
Group 1	125 Kbps	6	0	62 Kbps	58	0	68 Kbps	29	0
Group 3	125 Kbps	3	0	93 Kbps	26	0	114 Kbps	47	0
Group 6	125 Kbps	46	0	101 Kbps	72	0	110 Kbps	60	0

This table shows that our algorithm is much better for all three groups. The superiority is seen better as the deadlines are decreased.

CHAPTER 4

FIFO ANALYSIS

4.1 Overview

In this chapter, we analyze an unideal CAN network. In practical applications, due to several reasons FIFO queues are used in CAN controllers instead of priority queues. We study such kind of an unideal network which is a mixed CAN network consisting of both priority queued and FIFO queued CAN controllers. This chapter is sent to International Journal of Vehicle Design (IJVD) and accepted for that journal. It will be published on May. In Section 4.2, we define characteristics of FIFO CAN controllers. This chapter of the thesis is mainly based on [25] paper. Schedulability analysis of mixed CAN networks given in that paper is reviewed in Section 4.3. This chapter also suggests an improved schedulability analysis. We developed an algorithm which makes the FIFO analysis faster. Properties of our improved schedulability analysis algorithm is given in Section 4.4. Scheduling of FIFO messages is studied in Section 4.5 and scheduling algorithm for mixed CAN networks is implemented. Moreover, we further continue on FIFO analysis. In practical applications, it is very common to extend an existing message set by adding new messages. We want old messages to retain their original Id's, but new messages to get Id's accordingly. In literature, there are studies for extending ideal CAN networks. In Section 4.6, we suggest a new algorithm for extending mixed CAN networks. And finally, in Section 4.7, we demonstrate all of these cases.

4.2 Characteristics of FIFO Controllers

In the traditional schedulability analysis, it is assumed that the highest priority message in the queue always wins the arbitration. However, this might not be the case in practice. Due to several reasons, FIFO queues could be used instead of priority queues. First reason is that CAN systems are mass products, saving a very small amount from an item can be multiplied by millions of items. High-end cars today have about 70 CAN controllers ([18]). Therefore, because of the cost FIFO controllers can be preferred. Second reason is due to simplicity. In [36], it is stated that FIFO queues are simpler and provide faster queue management which improves the performance of the system. For example, Fujitsu MB90385/90387, Fujitsu 90390, Intel 87C196 (82527), Infineon XC161CJ/167 (82C900) components do not use priority queues, they transmit according to buffer number rather than message ID ([25]). Another usage of FIFO queues is in the gateways. In a gateway, many messages gather and the capacity of the gateway priority queue could be exceed, which is very probable. The most simple solution to this problem is using FIFO queues. In a car, not all controllers are FIFO, but FIFO and priority queued controllers may be used together which is a mixed CAN network.

Aside from schedulability analysis, scheduling of FIFO messages is investigated in this chapter. In the literature, there are many proposed scheduling algorithms. [13],[30],[15],[19] papers propose algorithms for priority queued CAN networks. For mixed CAN networks, in [11],[25] scheduling algorithms are developed. These algorithms are concerned with the scheduling of complete message set. In other words, all messages are initially unassigned and we assign completely new priorities to all of them.

CAN has been in use for three decades. Many different cars have been produced since then but message sets used in these cars have not changed much. When a new car is produced, car manufacturers use existing messages sets used in older cars. Because these message sets have been proven to work fine. Many documents are prepared for these message sets and reliability is tested in many cars. When a new feature is wanted to be added, they add new messages to the existing message sets but scheduling process does not start for all messages. This is called extending the

existing message set. This is a scenario frequently encountered in applications: set of messages is already assigned Id's and new messages have to be added to existing application.

Car manufacturers make the old messages keep their Id's and assign Id's to new messages. When a new message is added to the system, if the scheduling process starts for all messages and if new Id's are given to all messages, then there is no problem, many scheduling algorithms from literature can be used. For the extension case, how the Id's of new messages is chosen is a problem. For example, let the existing network consists of Id's 3, 6 and 9. In such case, the Id's 1,2,4,5,7,8 and all Id's larger than 9 are available for new messages. We have to choose appropriate Id's among these available gaps without violating schedulability. In the paper [39], a scheduling solution for extending existing message sets is proposed. This solution works only for ideal CAN networks which consists of only priority queued nodes. Using this work, we suggest a new algorithm for extending existing mixed CAN systems consisting of both FIFO and priority queued nodes. This is the main contribution of this paper.

4.3 Schedulability Analysis Algorithm

4.3.1 Notation and Scheduling Model

We consider messages that are transmitted on the Controller Area Network [40]. Each message m is specified by a tuple $m = (C_m, T_m, J_m, D_m, E_m)$, whereby C_m is the maximum transmission time, T_m is the message period , J_m is the release jitter, D_m is the deadline and E_m is the transmission deadline of m. C_m is evaluated according to (2.1), T_m represents the time between two message generations for periodic messages and the minimum inter-arrival time for sporadic messages, J_m represents the maximum time from the generation of m until its availability in the CAN device driver queue, D_m states the maximum allowable time between the generation of m until its successful arrival at a receiver node and $E_m = D_m - J_m$.

Regarding scheduling, we consider a CAN network with a set of messages $\mathcal{M}$. Since schedulability on CAN depends on the relative priority order of messages instead of

the absolute CAN ID [41, 19, 39], we introduce the priority order $o : \mathcal{M} \mapsto \{1, \ldots, |\mathcal{M}|\}$. Here, $o(m)$ represents the priority level of message $m \in \mathcal{M}$. The lowest priority level is $|\mathcal{M}|$ and the highest priority level is 1. For a message $m \in \mathcal{M}$, we write $\mathrm{lp}(m)$ for the set of lower-priority messages than m: $\mathrm{lp}(m) = \{m' \in \mathcal{M} | o(m') > o(m)\}$; $\mathrm{hp}(m)$ is the set of higher-priority messages than m: $\mathrm{hp}(m) = \{m' \in \mathcal{M} | o(m') < o(m)\}$. In addition, we use the *blocking time* $B_m = \max_{k \in lp(m)} C_k$, which represents the transmission time of the longest message in $lp(m)$.

Depending on the priority order o, each message $m \in \mathcal{M}$ has a *worst-case response time* (WCRT) denoted as R_m. We say that a priority order o is feasible for a message $m \in \mathcal{M}$ if $R_m \leq E_m$. If o is feasible for all $m \in \mathcal{M}$, o is called feasible. That is, the goal of message scheduling on CAN is determining a feasible priority order o.

The notation introduced in this section is summarized in Table 4.1.

Table4.1: Notation

General messages denoted as m	
C_m	Longest transmission time
D_m	Deadline
J_m	Release jitter
E_m	Transmission deadline
T_m	Period or minimum inter-arrival time
w_m	Queuing delay
R_m	Worst-case response time
$hp(m)$	Higher-priority messages than m
$lp(m)$	Lower-priority messages than m
B_m	Blocking time
o	Priority order
FQ messages	
G	Set of FIFO groups
G_m	FIFO group of message m
$M(m)$	Set of messages in the same FIFO group as m
L_m	Lowest-priority message in $M(m)$
E_m^{MIN}	minimum transmission deadline in $M(m)$
f_m	Buffering delay of message m
$f_{m,k}$	Buffering delay of message m for message k
C_m^{MAX}	Maximum transmission time in $M(m)$
C_m^{MIN}	Minimum transmission time in $M(m)$
C_m^{SUM}	Accumulated transmission time in $M(m)$

4.3.2 Worst-case Response Time Computation

The most commonly used scheduling model for CAN [19] assumes that the the device driver of a CAN node implements a priority queue (PQ) such that always the highest priority available message occupies the head of the PQ and enters arbitration. Nevertheless, as is pointed out by [36, 25], not all CAN device driver implementations comply with this assumption and use FIFO queues (FQ) instead. In this case, each message has to wait until it becomes the oldest message in the FQ in order to enter arbitration. As a result, each FIFO-queued message may experience an additional *buffering delay* when queued behind lower-priority messages, leading to an increased WCRT compared to the analysis by [19].

In this paper, the schedulability analysis and priority assignment for messages on CAN networks that include both PQ and FQ nodes. Since our research is based on WCRT computation by [25], we briefly summarize this computation. We first introduce additional notation in order to describe CAN nodes with FQs. We consider a set G of FIFO groups. We write $G_m \in G$ for the FIFO group of message $m \in \mathcal{M}$ and $M(m)$ for the set of messages that enter the same FQ as m. L_m is the lowest-priority message in $M(m)$. The maximum and minimum transmission time of messages in $M(m)$ is C_m^{MAX} and C_m^{MIN}, respectively, and the sum of all transmission times of messages in $M(m)$ is evaluated as $C_m^{\text{SUM}} = \sum_{k \in M(m)} C_k$. FQ messages potentially experience an additional buffering delay before reaching the head of the FQ. The maximum buffering delay of FQ message $m \in \mathcal{M}$ is written as f_m.

The notation introduced in this section is summarized in Table 4.1.

4.3.2.1 WCRT Computation for PQ Messages

We recall the WCRT computation for a PQ message $m \in \mathcal{M}$ from [25]. To this end, first the *queuing delay* w_m is evaluated by the following iteration

$$w_m^{n+1} = \max(B_m, C_m) + \sum_{k \in hp(m)} \left\lceil \frac{w_m^n + J_k + f_{k,m} + \tau_{\text{bit}}}{T_k} \right\rceil C_k \tag{4.1}$$

Hereby, $f_{k,m}$ represents the effect of the buffering delay of the higher-priority message $k \in \text{hp}(m)$ on the queuing delay of m. It holds that $f_{k,m} = f_k$ or $f_{k,m} = 0$ as is discussed

in Section 4.4. The iteration in (4.1) starts with $w_m^0 = C_m$ and terminates when either $w_m > E_m$ (deadline violation) or $w_m^{n+1} = w_m^n$ (convergence). In the first case, the system is not schedulable. In the second case, the WCRT of m is determined as

$$R_m = w_m^{n+1} + C_m. \tag{4.2}$$

4.3.2.2 WCRT Computation for FQ Messages

For each FQ message $m \in \mathcal{M}$, the queuing delay w_m is evaluated as follows

$$w_m^{n+1} = \max(B_{L_m}, C_m^{\text{MAX}}) + C_m^{\text{SUM}} - C_m^{\text{MIN}} + \\ + \sum_{k \in (hp(L_m) \setminus M(m))} \left\lceil \frac{w_m^n + J_k + f_{k,m} + \tau_{\text{bit}}}{T_k} \right\rceil C_k \tag{4.3}$$

We note that this computation is FIFO symmetric in the sense that it is assumed that all messages $k \in M(m)$ obtain the same queuing delay. Accordingly, the computation considers the blocking time B_{L_m} of the lowest-priority message in $M(m)$, the maximum transmission time C_m^{MAX} and the maximum accumulated transmission time $C_m^{\text{SUM}} - C_m^{\text{MIN}}$ of FQ messages in front of the tail of the FQ. Again, $f_{k,m}$ represents the effect of the buffering delay of the higher-priority message $k \in \text{hp}(m)$ on the queuing delay of m with $f_{k,m} = f_k$ or $f_{k,m} = 0$ as is discussed in Section 4.4. The iteration in (4.3) starts with $w_m^0 = \max(B_{L_m}, C_m^{\text{MAX}}) + C_m^{\text{SUM}} - C_m^{\text{MIN}}$ and terminates when either $w_m > E_m^{\text{MIN}} = \min_{k \in M(M)}(E_k)$ (deadline violation of at least one message in $M(m)$) or $w_m^{n+1} = w_m^n$ (convergence). In the first case, the system is not schedulable. In the second case, the WCRT of m is determined as

$$R_m = w_m^{n+1} + C_m^{\text{MIN}} \tag{4.4}$$

and the maximum buffering delay of m is computed as

$$f_m = w_m^{n+1} = R_m - C_m^{\text{MIN}}. \tag{4.5}$$

4.3.2.3 Schedulability Analysis Algorithm

An algorithm for evaluating the WCRTs in (4.2) and (4.4) is proposed by [25] in Algorithm 6. Since we suggest an improvement of this algorithm in Section 4.4, we

briefly discuss the main features of the algorithm in [25]. It is noted that there can be a circular dependency when evaluating (4.3) for FQ messages of FIFO groups with interleaved priority levels. To illustrate this dependency, let k and m be two messages whose FIFO groups G_k and G_m have interleaved priority levels. Then, $f_{k,m} = f_k$ is needed for the computation of w_m and $f_{m,k} = f_m$ is needed for the computation of w_k. In order to break this circular dependency, [25] observe that buffering delay values according to (4.5) and (4.3) are monotonically increasing with each iteration. Hence, they perform the WCRT computation in a loop over all messages $m \in \mathcal{M}$ that is repeatedly restarted. To this end, the computation according to (4.1) and (4.3) is performed for a fixed value of the buffering delay for each message. If an inconsistency is detected, that is, $f_m < w_m$ for any message $m \in \mathcal{M}$, the buffering delay is updated to $f_m = w_m$ and the computation is restarted. The algorithm terminates if $f_m = w_m$ is obtained for all FQ messages.

input : Full message set with corresponding priority assignment $o(m)$ for each message m

output: WCRT R_m for each message m

```
1  Initialize f_g := 0 for all g ∈ G
2  repeat=true;
3  for all priority levels l starting from the highest level do
4      Let m be the message with p_m = l
5      if m is a FIFO queued message then
6          Compute R_m according to (4.3) and (4.4) if R_m > D_m then
7              return unschedulable;
8          end
9          if f_{m,k} < w_m then
10             f_{m,k} = w_m repeat=true;
11         end
12     end
13     else
14         Compute R_m according to (4.1) if R_m > D_m then
15             return unschedulable;
16         end
17     end
18 end
19 return schedulable;
```

Algorithm 6: WCRT computation for CAN networks with FIFO queues.

4.4 Improved Algorithm for Schedulability Analysis with FIFO Queues

In this section, we present our improved algorithm for the schedulability analysis of CAN networks with PQs and FQs. Section 4.4.1 discusses special cases that simplify the WCRT computation and Section 4.4.2 states our improved algorithm based on these special cases. An illustrative example is given in Section 4.4.3.

4.4.1 Special Cases

The evaluation of the queuing delay in (4.1) and (4.3) for a message $m \in \mathcal{M}$ requires the knowledge of the buffering delay $f_{k,m}$ for each higher-priority message $k \in hp(m)$. As is stated by [25], $f_{k,m} = 0$ for all PQ messages. In addition, in certain special cases, $f_{k,m} = 0$ also for FQ messages. Otherwise $f_{k,m} = f_m$. We next rigorously investigate the special cases in which $f_{k,m} = 0$ is valid for FQ messages with the aim of simplifying the algorithm by [25].

First, we define the notion of *spanning* for FIFO groups.

Definition 4.4.1 *(a) Consider a PQ message $m \in \mathcal{M}$ and a FIFO group G_k for some FQ message $k \in \mathcal{M}$. We say that G_k spans m if $hp(m) \cap M(k) \neq \emptyset$ and $lp(m) \cap M(k) \neq \emptyset$.*

(b) Consider an FQ message $m \in \mathcal{M}$ and a FIFO group G_k. We say that G_k spans m if $hp(L_m) \cap M(k) \neq \emptyset$ and $lp(L_m) \cap M(k) \neq \emptyset$.

In words, (a) states that a FIFO group G_k spans a PQ message m if the priority level of m is between the highest and the lowest priority level of G_k. Similarly, (b) states that a FIFO group G_k spans a FQ message m if the lowest priority level L_m of G_m is between the highest and the lowest priority level of G_k.

We illustrate the concept of spanning FIFO groups using the example setting in Fig. 4.1. In this example, there are 4 FIFO groups FIFO0, FIFO1, FIFO2, FIFO3 and two PQ messages PQ2 and PQ5. The priorities of the respective messages increase according to the arrow in the figure and the priority levels of message in each FIFO group lie between the indicated upper and lower bound. It holds that the PQ message j_2 = PQ2 is spanned by the FIFO group FIFO1 and the PQ message j_5 = PQ5 is spanned by the FIFO group FIFO4. Likewise, all FQ messages of FIFO0 (such as j_0) are spanned by FIFO1, all FQ messages of FIFO1 (such as m_1) are spanned by FIFO 3 and all FQ messages of FIFO3 (such as j_3) are spanned by FIFO4. The FQ messages of FIFO4 are not spanned by any FIFO group.

Using the idea of spanning FIFO groups, several facts regarding the evaluation of the buffering delay $f_{k,m}$ in (4.1) and (4.3) are stated by [25] and summarized in Proposi-

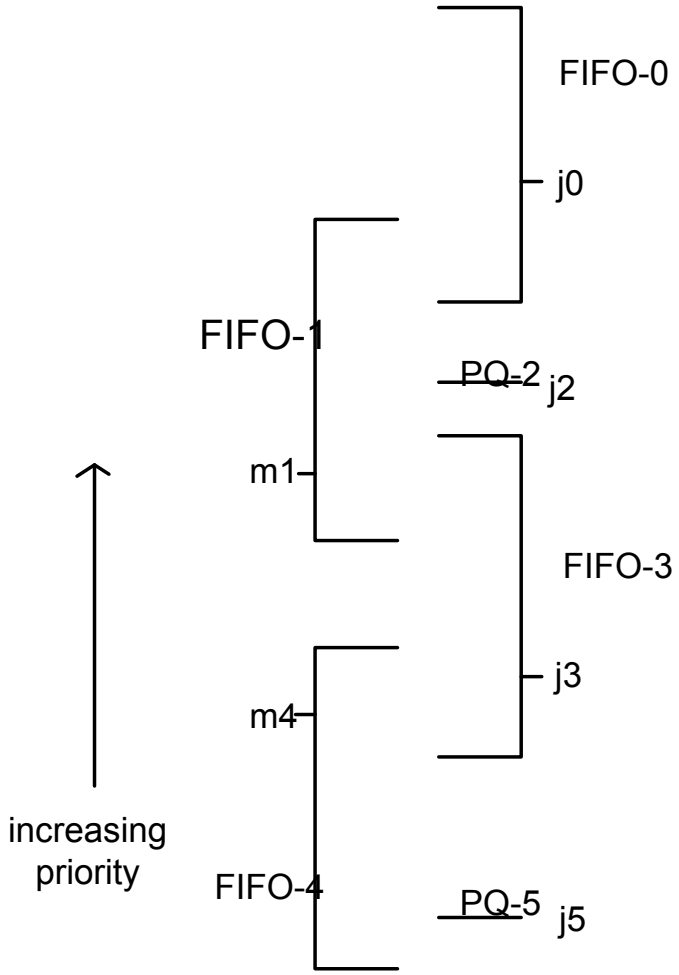

Figure 4.1: Spanning and Non-Spanning Groups

tion 4.4.2.

Proposition 4.4.2 *(a) Let m be a PQ message and G_k be a FIFO group such that $hp(m) \cap M(k) \neq \emptyset$. $f_{k,m} = f_k$ if G_k spans m. Otherwise, $f_{k,m} = 0$.*

(b) Let m be an FQ message and G_k be a FIFO group such that $hp(m) \cap M(k) \neq \emptyset$. $f_{k,m} = f_k$ if G_k spans m. Otherwise, $f_{k,m} = 0$.

In words, (a) states that the buffering delay $f_{k,m}$ for a PQ message m evaluates to zero for all messages k that belong to FIFO groups whose priority levels lie entirely above the priority level of m. In Fig. 4.1, the WCRT of PQ2 is evaluated with $f_{k,m} = 0$ for all messages of FIFO0 and the WCRT of PQ5 is evaluated with $f_{k,m} = 0$ for all messages k of FIFO3, FIFO1 and FIFO0. Similarly, (b) states that the buffering delay $f_{k,m}$ for an

FQ message m evaluates to zero for all messages k that belong to FIFO groups whose priority levels lie entirely above the priority level of the lowest priority message L_m of G_m. In Fig. 4.1, the WCRT of m_4 is evaluated with $f_{k,m} = 0$ for all messages k of FIFO3, FIFO1 and FIFO0 and the WCRT of m_1 is evaluated with $f_{k,m} = 0$ for all messages k of FIFO0.

We demonstrate the implications of Proposition 4.4.2 using Fig. 4.1. Considering the PQ message j_2, 1. implies that $f_{k,m} = 0$ for all messages k of FIFO0, whereas $f_{k,m} = f_k$ for all higher-priority messages of FIFO1 since FIFO1 spans j_2. Similarly, when calculating the WCRT of the PQ message j_5, $f_{k,m} = 0$ for all messages k of FIFO0, FIFO1 and FIFO3 (not spanning). Only $f_{k,m} = f_k$ for the higher-priority FQ messages k of FIFO4. For the WCRT of the FQ message j_3, $f_{k,m} = 0$ for the FQ messages in FIFO0 and FIFO1 (non-spanning). Only $f_{k,m} = f_k$ for the higher-priority messages k in FIFO4 (spanning).

4.4.2 Improved Algorithm

Based on the properties stated in Proposition 4.4.2, we obtain the following new result about the WCRT computation.

Proposition 4.4.3 *(a) Let m be a PQ message and $k \in hp(m)$ be an FQ message. Then, $f_{k,m} = f_{L_k} = w_{L_k}^{n+1}$ if $L_k \in lp(m)$. Otherwise, $f_{k,m} = 0$.*

(b) Let m be an FQ message and $k \in hp(m) \setminus M(m)$ be an FQ message of another FIFO group. Then, $f_{k,m} = f_{L_k} = w_{L_k}^{n+1}$ if $L_k \in lp(L_m)$. Otherwise, $f_{k,m} = 0$.

Proof.

(a) Let m and k be as specified in Proposition 4.4.3 (a). If $L_k \in \mathrm{lp}(m)$, it holds that $\mathrm{lp}(m) \cap M(k) \neq \emptyset$. Also, $\mathrm{hp}(m) \cap M(k) \neq \emptyset$ since $k \in \mathrm{hp}(m)$. Hence, by Definition 4.4.1 (a), G_k spans m. Then, by Proposition 4.4.2 (a), $f_{k,m} = f_k$ and with (4.5), $f_k = w_k^{n+1}$. Considering that our analysis is FIFO symmetric, it holds that $w_k^{n+1} = w_{L_k}^{n+1}$, that is, $f_{k,m} = w_{L_k}^{n+1}$. If $L_k \notin \mathrm{lp}(m)$, Definition 4.4.1 (a) implies that G_k does not span m. Hence, $f_{k,m} = 0$ with Proposition 4.4.2 (a).

(b) Let m and k be as specified in Proposition 4.4.3 (b). If $L_k \in \text{lp}(L_m)$, it holds that $\text{lp}(L_m) \cap M(k) \neq \emptyset$. Also, $\text{hp}(L_m) \cap M(k) \neq \emptyset$, since $k \in \text{hp}(m)$. By Definition 4.4.1 (b), G_k spans m and by Proposition 4.4.2 (b), $f_{k,m} = f_k$. Then, $f_k = w_k^{n+1} = w_{L_k}^{n+1} = f_{L_k}$ by (4.5) and FIFO symmetry. If $L_k \notin \text{lp}(L_m)$, G_k does not span m with Definition 4.4.1 (b). Then, Proposition 4.4.2 (b) implies that $f_{km} = 0$.

■

In words, Proposition 4.4.2 states that non-zero buffering delays $f_{k,m} \neq 0$ for the queuing delay computation of a message m only appear for higher-priority messages k of FIFO groups whose lowest-priority message L_k has a lower priority than m. In addition, it then holds that the buffering delay is equal to the queuing delay of L_k: $f_{k,m} = w_{L_k}^{n+1}$.

Considering the result in Proposition 4.4.2, the circular dependency identified by [25] can be removed by starting the WCRT computation from the lowest-priority message. In that case, the required non-zero buffering delays $f_{k,m} = w_{L_k}^{n+1}$ for the evaluation of (4.1) and (4.3) are always known since $L_k \in \text{lp}(m)$ and $w_{L_k}^{n+1}$ is evaluated before w_k^{n+1}. Based on this discussion, we present our improved algorithm for the WCRT computation on CAN networks with PQs and FQs.

Algorithm proceeds from lowest priority level to the highest priority level. If the analyzed message is a PQ message, (4.1) and (4.2) are applied using the evaluation of $f_{k,m}$ as determined in Proposition 4.4.2 (line 5). If the analyzed message is an FQ message, then we check whether it is the lowest priority message on its FIFO group (line 8). In the positive case, (4.3) is evaluated to determine the queuing delay using $f_{k,m}$ as determined in Proposition 4.4.2 (line 9). The buffering delay of the FIFO group is then equal to the found queuing delay (line 10) and the WCRT follows from (4.4) (line 11). In the negative case, the WCRT of the considered message is equal to the previously computed WCRT of the lowest-priority message in the FIFO group (line 14). It is readily observed that the proposed algorithm improves the schedulability analysis in [25]. First, the WCRT computation only needs to be evaluated for all PQ messages and one FQ message in each FIFO group. In [25], the WCRT computation has to be performed for all messages. Second and more importantly, our algorithm does not need any restart. That is, whereas our algorithm performs the

WCRT computation for each PQ message and the lowest-priority messages in each FIFO group only once, the WCRT computation for each message potentially has to be performed many times until convergence in [25]. As is shown in Section 4.7.1, this leads to significant computational savings of the proposed algorithm.

4.4.3 Illustrative Example

Table4.2: Example Message Set

Message	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}	m_{12}
Node	FQ1	PQ2	FQ3	FQ1	FQ1	FQ3	FQ4	FQ3	FQ4	PQ5	PQ5	FQ4
Period	20	20	20	20	20	20	20	20	20	20	20	20
Deadline	20	20	20	20	20	20	20	20	20	20	20	20
WCRT	6	3	10	6	6	10	13	10	13	13	14	13

For better understanding, we apply Algorithm 7 to an example message set with 12 messages. The message properties are summarized in Table 4.2, whereby it is assumed that the priority level of each message is given by its name. For simplicity, we assume a bus speed of 125 Kbps and a payload size of 7 bytes for each message (C_m = 1 ms for each message). The priority order is displayed in Fig. 4.2.

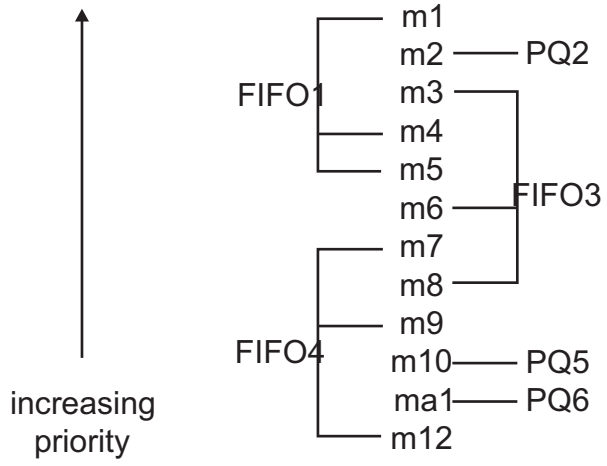

Figure 4.2: Spanning and Non-Spanning Groups

The schedulability analysis starts with the lowest-priority message m_{12}. It is the FQ message with the lowest priority in the group FIFO4. Therefore, the buffering delay

$f_{k,\mathrm{m12}}$ is zero for all other FQ messages since no FIFO group spans m_{12}. We obtained $w_{m_{12}}^{n+1} = 12$ in line 9 of Algorithm 7 and $R_{m12} = 13$ (line 11). Then buffering delay of FQ4 is $f_{m_{12}} = w_{m_{12}}^{n+1} = 12$ (line 10). When calculating the WCRT of the PQ message m_{11}, the buffering delays of messages in FQ3 and FQ1 are zero because these groups do not span m_{11}. Nevertheless, the buffering delay of messages in FQ4 is $f_{m_{12}} = 12$ since FQ4 spans m_{11}. The WCRT of m_{11} is determined as $R_{m_{11}} = 14$ (line 5). The WCRT calculation for m_{10} is very similar to that of m_{11}. Next, the WCRT of m_9 need not be calculated since m_9 is in the same FIFO group as m_{12}. Because of FIFO symmetry, $R_{m_9} = R_{m_{12}} = 13$ (line 14). m_8 is the lowest-priority message of FQ3 and is spanned only by FQ4. Hence, The WCRT computation for m_8 considers that $f_{m_8,m_7} = f_{m_{12}} = 13$ and $f_{m_8,k} = 0$ for all other messages. The resulting WCRT is $R_{m_8} = 10$ (line 11) and the buffering delay of FQ3 is $f_{m_8} = 9$ (line 10). The remaining WCRTs are calculated similarly. The results are listed in Table 4.2.

For comparison, we also applied the algorithm by [25] and obtained the same WCRT values for this example. Nevertheless, the algorithm by [25] needs 24 iterations due to frequent restarts, whereas our algorithm completes the WCRT computation in only 6 iterations. We note that this difference becomes an important parameter for larger example sets as is shown in Section 4.7.1.

input : Message set $\mathcal{M}$ with priority assignment $o(m)$ for each message $m \in \mathcal{M}$

output: WCRT R_m for each message $m \in \mathcal{M}$

Initialize $f_{L_k} := 0$ for all lowest-priority messages L_k of FiFO groups
for *all priority levels l starting from the lowest level* **do**
    Let m be the message with $o(m) = l$
    if *m is a PQ message* **then**
        Compute R_m using (4.1) and (4.2) with $f_{k,m} = f_{L_k}$ for all FQ messages such that $\text{hp}(m) \cap M(k) \neq \emptyset$ and G_k spans m. $f_{k,m} = 0$ for all remaining messages in $\text{hp}(m)$.
    end
    else
        if $m = L_m$ *is the lowest priority message in* $M(m)$ **then**
            Compute w_m^{n+1} using (4.3) with $f_{k,m} = f_{L_k}$ for all FQ messages such that $\text{hp}(m) \cap M(k) \neq \emptyset$ and $M(k)$ spans m. $f_{k,m} = 0$ for all remaining messages in $\text{hp}(m)$.
            Set $f_{L_m} := w_m^{n+1}$
            Compute $R_m := w_m^{n+1} + C_m^{\text{MIN}}$ (using (4.4))
        end
        else
            $R_m := R_{L_m}$ (using FIFO symmetry)
        end
    end
end

Algorithm 7: Improved WCRT computation for CAN networks with FIFO queues.

4.5 FIFO Scheduling

The idea behind the scheduling algorithm is that we assign messages to the available priority levels. This scheduling algorithm uses the schedulability test in each step. It assigns a message to a priority level and then tests whether this message is schedulable at that priority or not. If schedulable, then leave it and go to another priority level. If not schedulable, then try another message for that priority level and test it again. Similar to the schedulability algorithm, there are some differences between priority queued and FIFO queued messages. For FIFO queued messages, a special case exists. For a priority level, if we assign a FIFO message to that level, then adjacent priority levels should be assigned to the other messages at the same node. Starting from the lowest priority level, we try to match the message to the levels. In which order, we try messages to the priority levels also matter. In the paper, in default, given messages are sorted in smallest deadline first order. We try to match the largest deadlined message to the lowest priority level. The pseudo-code is given in Algorithm 8.

input : Message sets S with unassigned priorities

output: Priority Ordering o

```
1  Initialize Message Sets H = S , L = emptySet;
2  for each priority band k, lowest first do
3      for each message m in S do
4          if m is schedulable in priority band k according to schedulability test
           described in Algorithm 6 with all unassigned priority-queued messages
           / other FIFO groups assumed to be in higher priority bands then
5              Assign m to that priority level, o(m) = k;
6              H = H/m, L = LUm;
7              if m is part of a FIFO group then
8                  assign all other messages in the FIFO group to adjacent
                   priorities within priority band k;
9              end
10             break(break to outer loop);
11         end
12     end
13 end
```

Algorithm 8: Priority Assignment for CAN systems with FIFO nodes.

4.6 Message Set Extension

4.6.1 Motivation

The previous section develops an improved algorithm for the schedulability analysis of CAN networks with PQs and FQs. For such analysis, it is assumed that all messages on the CAN network are known and a priority level is assigned to each message. In addition, [25] considers the case where a set of messages without a priority assignment is given. In this case, an algorithm for the computation of such priority assignment is provided. A characteristic of this algorithm is that the resulting priority assignment is *FIFO-adjacent*, that is, messages in the same FIFO group are assigned adjacent priority levels. Nevertheless, practical applications do not only require the schedulability analysis for an existing message set with given priority levels or the computation of priority levels for an entire message set. A frequently encountered situation is that an existing message set with given CAN IDs is extended by new messages [39]. In this case, the task is to place the CAN IDs of the new messages between the CAN IDs of the existing messages, while maintaining schedulability of the existing messages and achieving schedulability for the new messages. This problem is solved by a priority assignment algorithm for the case of CAN networks with PQs by [39]. In this thesis, we develop an algorithm that performs the priority assignment for new messages that are added to an existing CAN network with both PQs and FQs.

4.6.2 Extension Algorithm

We assume that the set of messages $\mathcal{M}$ is composed of a set of fixed messages $\mathcal{F}$ and a set of new messages $\mathcal{N}$. We write $\mathcal{F} = \{F_1, \ldots, F_{|\mathcal{F}|}\}$ and assume that the messages in $\mathcal{F}$ are sorted by decreasing priority (increasing CAN ID) such that F_1 has the highest priority and $F_{|\mathcal{F}|}$ has the lowest priority. Using $\mathrm{id} : \mathcal{F} \mapsto \mathbb{N}$ to define the fixed CAN ID $\mathrm{id}(F_i)$ of a message $F_i \in \mathcal{F}$, it is observed from the fixed ID assignment for messages in $\mathcal{F}$ that the new messages in $\mathcal{N}$ can only obtain IDs in the *ID gaps* between the messages with fixed ID. We write $g_i = \mathrm{id}(F_{i+1}) - \mathrm{id}(F_i) - 1$ for the gap between message F_i and F_{i+1} for $i = 1, \ldots, |\mathcal{F}| - 1$. In addition, we introduce $g_0 = \mathrm{id}(F_1)$ and $g_{|\mathcal{F}|} = 2^q - \mathrm{id}(F_{|\mathcal{F}|}) - 1$ ($q = 11$ or $q = 29$) for the gaps before the first and after the last

CAN message with fixed ID. For notational convenience, we further divide the set $\mathcal{M}$ in PQ messages $\mathcal{M}^{\text{PQ}}$ and FQ messages $\mathcal{M}^{\text{FQ}}$. Likewise, we write $\mathcal{N}^{\text{PQ}} = \mathcal{N} \cap \mathcal{M}^{\text{PQ}}$, $\mathcal{N}^{\text{FQ}} = \mathcal{N} \cap \mathcal{M}^{\text{FQ}}$, $\mathcal{F}^{\text{PQ}} = \mathcal{F} \cap \mathcal{M}^{\text{PQ}}$ and $\mathcal{F}^{\text{FQ}} = \mathcal{F} \cap \mathcal{M}^{\text{FQ}}$. Finally, for a FIFO group $G_k \in G$, we write $\mathcal{M}_{G_k}$ for the set of messages that belongs to G_k. The notation is summarized in Table 4.3.

Table4.3: Notation for Message Set Extension

Notation	Explanation
$\mathcal{M}^{\text{PQ}}$	PQ messages
$\mathcal{M}^{\text{FQ}}$	FQ messages
$\mathcal{F} = \{F_1, \ldots, F_{\lvert\mathcal{F}\rvert}\} \subseteq \mathcal{M}$	messages with fixed CAN ID
$\mathcal{N} \subseteq \mathcal{M}$	new messages
$\mathcal{F}^{\text{PQ}}$	PQ messages with fixed CAN ID
$\mathcal{F}^{\text{FQ}}$	FQ messages with fixed CAN ID
$\mathcal{N}^{\text{PQ}}$	new PQ messages
$\mathcal{N}^{\text{FQ}}$	new FQ messages
g_i	gap between messages with fixed CAN ID
$\mathcal{M}_{G_k}$	messages that belong to FIFO group G_k

Using the previously introduced notation, the main objective of this section is to determine a schedulable priority order $o : \mathcal{M} \mapsto \mathbb{N}$ such that

- Messages $F_i \in \mathcal{F}$ obtain priority levels $o(F_i)$ such that they can keep their fixed CAN IDs $\text{id}(F_i)$,
- Messages $N \in \mathcal{N}$ obtain priority levels $o(N)$ in the gaps between the fixed CAN IDs.

The difficulty of the stated problem is that, in principle, any new messages could be assigned any priority level in the available gaps between fixed messages. That is, considering that there are $|\mathcal{N}|$ new messages and $\mathcal{F} + 1$ gaps, there are up to $\binom{|\mathcal{F}|+1}{|\mathcal{N}|}$ possible priority assignments. Moreover, the presence of FQ messages with potentially non-adjacent priority levels and the possible lack of gaps that are adjacent to FQ messages make the priority assignment difficult.

In order to solve the stated problem, we propose a priority assignment algorithm that tries to place each new message $N \in \mathcal{N}$ in the most appropriate gap between fixed

messages. The algorithm starts with the full set of messages $\mathcal{W} = \mathcal{F} \cup \mathcal{N}$ from the lowest priority level. It keeps the boolean variable FIFO to indicate if some FIFO group spans the current priority level, the variable g_{cur} to indicate how many new messages fit in the current gap and the variable G^{un} for the FIFO groups with unassigned messages. The basic strategy of our algorithm is to

1. prefer assigning PQ messages to lower-priority levels in order to achieve smaller WCRTs and hence buffering delay for FQ messages,
2. assign the lowest-priority message of any FIFO groups such that the buffering delay of the respective FIFO groups is minimized,
3. assign FQ messages of FIFO groups with a lower-priority lowest-priority message first.

Since our algorithm comprises different cases, we present it in 4 parts. Algorithm 9 contains the main loop with 3 different cases that are addressed in Algorithm 10, 11 and 12, respectively.

The main loop in Algorithm 9 inspects each priority-level l starting from the lowest priority-level $l = |\mathcal{W}|$ (line 1). In Case 1 (line 2), there is space for new messages in the current gap and no FIFO group spans priority-level l (`FIFO` = false). In Case 2 (line 5), there is space for new messages in the current gap and at least one FIFO groups spans priority-level l. In Case 3 (line 8), the current gap is fully occupied.

Algorithm 10 is applied in Case 1. The algorithm tries to find an appropriate new message for the available gap. Hereby, S denotes a set of identified candidate messages that is initially empty (line 1). Considering that no FIFO group spans the current priority level l, unassigned PQ messages are tried first (line 2) and all feasible such messages are inserted in S (line 5). If at least one feasible candidate PQ message is found (line 8), the candidate with the largest value of C_N/T_N is chosen since the remaining candidates (to be assigned to higher priority levels) potentially cause a smaller interference on the WCRT of other messages. Line 10 performs the priority assignment to N and updates the variables of the main loop. Then, the algorithm returns to the main loop (line 11). If no candidate PQ message is found, the next choice is to assign the current priority-level l to a new FQ message. Since no FIFO

input : $f_{G_k} = 0$ for all $G_k \in G$; $\mathcal{W} := \mathcal{F} \cup \mathcal{N}$; $l := |\mathcal{W}|$; `FIFO` := **false**;
$G^{\text{new}} := \cup_{m \in \mathcal{N}^{\text{FIFO}}} G_m$; $n := |\mathcal{F}|$; $g_{\text{cur}} := g_n$

output: Priority assignment o

```
1  while l > 0 do
2      if g_cur > 0 ∧ FIFO = false then
3          Case 1: Apply Algorithm 10 to assign an appropriate message to
                   priority level l.
4      end
5      else if g_cur > 0 ∧ FIFO = true then
6          Case 2: Apply Algorithm 11 to assign an appropriate message to
                   priority level l.
7      end
8      else
9          Case 3: Apply Algorithm 12 to assign an appropriate message to
                   priority level l.
10     end
11 end
```

Algorithm 9: Message set extension algorithm: main loop.

group spans l in Case 1, this FQ message is the lowest-priority message in its FIFO group. Line 13 checks all unassigned FIFO groups and line 14 determines the new FQ message with the smallest WCRT in its group. If this FQ message is feasible, it is inserted as candidate in S (line 16). If at least one candidate FQ message is found, the candidate N with the smallest WCRT obtains priority-level l (line 20). Since N is the lowest-priority message in its FIFO group, also the buffering delay is set as $f_N = w_N^{n+1}$. If no FQ message is found, this implies that the current gap cannot be used for new messages. Hence, priority-level l must be used for the next fixed message F_n. If F_n is a PQ message (line 24), it obtains $o(F_n) = l$ if it is feasible and the main loop moves to the next gap (line 27). The same procedure is applied if F_n is an FQ message (line 34 to 39). Hereby, it has to he noted that, in both cases, the buffering delay of higher-priority messages is zero since no FIFO group spans l. In addition, the evaluation is only performed if $\sum_{i=1}^{n-1} g_i \geq |\mathcal{W}|$, that is, there are enough gaps for the unassigned new messages. If none of the evaluations in Algorithm 10 leads to a positive result, the message set is deemed unschedulable.

In Case 2 of Algorithm 9, there is space in the current gap and the current priority-level l is spanned by at least one FIFO group. In principle, any FQ message N of any FIFO group G_k such that $o(L_k) < l$ has already be assigned can be chosen since $R_N \leq E_N^{\min}$ according to line 36 in Algorithm 10. In Algorithm 11, we suggest to choose the message N that belongs to the FIFO group G_k with the lowest priority $o(L_k)$ and with the largest C_N/T_N. The aim is to minimize the number of messages spanned by FIFO group G_k and to keep the interference of unassigned messages of G_k small. If all messages of G_k are assigned (line 4), G_k is removed from the set of new FIFO groups G^{new}. Further, if all FIFO groups are either fully assigned or unassigned (line 7), FIFO becomes false, since the next priority-level is not spanned by any FIFO

group.

1 Find $G_k \in G^{\text{new}}$ such that $o(L_k) > o(L_{k'})$ for any other $G_{k'} \in G^{\text{new}}$
2 Choose $N \in \mathcal{N}^{\text{FQ}} \cap \mathcal{M}_{G_k} \cap \mathcal{W}$ with the largest value of C_N/T_N
3 $o(N) := l$; $l := l - 1$; $\mathcal{W} := \mathcal{W} \setminus \{N\}$; $g_{\text{cur}} := g_{\text{cur}} - 1$
4 **if** $\mathcal{N}^{\text{FIFO}} \cap \mathcal{M}_{G_k} \cap \mathcal{W} = \emptyset$ **then**
5 | $G^{\text{new}} := G^{\text{new}} \setminus \{G_k\}$
6 **end**
7 **if** $\cup_{G_k \in G^{\text{new}}} \mathcal{M}_{G_k} \cap \mathcal{W} = \cup_{G_k \in G^{\text{new}}} \mathcal{M}_{G_k}$ **then**
8 | `FIFO` := **false**
9 **end**
10 **continue** with main loop

Algorithm 11: Message set extension algorithm Case 2: at least one FIFO group spans the current priority-level.

In case 3 of Algorithm 9, there is no space in the current gap. In that case, it needs to be checked if the next fixed message F_n is feasible at the current priority-level l. This check is performed for PQ messages in line 1 to 10 and for FQ messages in line 11 to 26. If the check is positive, priority-level l is assigned to F_n (line 4 or line 20).

Otherwise, the message set is deemed unschedulable.

```
1  if F_n ∈ M^PQ then
2      Compute R_{F_n} using (4.1) and (4.2)
3      if R_{F_n} ≤ E_{F_n} then
4          o(F_n) := l; l := l − 1; W := W \ {F_n}; n := n − 1; g_cur := g_n
5          continue with main loop
6      end
7      else
8          return not schedulable
9      end
10 end
11 else
12     if F_n = L_{F_n} then
13         Compute R_{F_n} using (4.3) and (4.4)
14         f_{F_n} := w_{F_n}
15     end
16     else
17         R_{F_n} = R_{L_{F_n}}
18     end
19     if R_{F_n} ≤ E^min_{F_n} then
20         o(F_n) := l; l := l − 1; W := W \ {F_n}; n := n − 1; g_cur := g_n
21         continue with main loop
22     end
23     else
24         return not schedulable
25     end
26 end
```

Algorithm 12: Message set extension algorithm Case 3: there is no space in the current gap.

4.6.3 Message Set Extension Example

We consider the existing message set $\mathcal{F}$ as given in Table 4.4 and the new messages in Table 4.5. Here, the CAN IDs of the existing messages are assigned such that there is a gap of $g_i = 1$, $i = 0,\ldots,11$ between any of the neighboring messages. The gap after the last existing message is $g_{12} = 2^q - 24$. Two of the new messages belong to a new node FQ6 with a FIFO queue and the two remaining messages are added to the node PQ5 with a PQ.

Table4.4: Existing Message Set $\mathcal{F}$

Message	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}
CAN ID	1	3	5	7	9	11	13	15	17	19	21	23
Queue	FQ1	PQ2	FQ3	FQ1	FQ1	FQ3	FQ4	FQ3	FQ4	PQ5	PQ5	FQ4
Period	20	20	40	20	20	40	40	40	40	20	20	40
Deadline	20	20	40	20	20	40	40	40	40	20	20	40

Table4.5: Extension Set $\mathcal{N}$

Message	A	B	C	D
Node	FQ6	FQ6	PQ5	PQ5
Period	10	40	20	10
Deadline	10	40	20	10

We apply Algorithm 9 to this example. Since $|\mathcal{F}| + |\mathcal{N}| = 16$, the algorithm starts with $l = 16$, $n = 12$ and $g_{\text{cur}} = g_{12}$. This priority level is not spanned by any FIFO group and $g_{\text{cur}} > 0$,which corresponds to Case 1 in Algorithm 9. Hence, first the PQ messages C and D are tried (line 2 in Algorithm 10. Message D violates the deadline, but message C fits. Hence, $o(\text{C}) = 16$ is assigned. For $l = 15$, again, the PQ message D is tried and still violates the deadline. According to line 13 in Algorithm 10, the WCRTs of the new FQ messages A and B are computed. Since none of these messages is feasible, the current fixed message F_{12} should be assigned to $l = 15$. Using line 34 in Algorithm 10, it is found that F_{12} is feasible. Hence, $o(F_{12}) = 15$ and the algorithm moves to the next gap $g_{\text{cur}} = g_{11}$. The algorithm then determines that no new messages can be placed in the gaps between the fixed messages F_7 to F_{11} and the priority levels 10 to 14 are assigned to these messages. At $l = 9$, the new FQ

message B is feasible (line 13 to 23 in Algorithm 10) such that o(B) = 9. Since the unassigned new FQ message A obtains a higher priority than B, FIFO = true is set in line 20 of Algorithm 10. Although it would be desired to place the new FQ message A adjacent to B, this is not possible since the gap between F_6 and F_7 is already filled. That is, $o(F_7) = 8$ and it is found that o(A) = 7 is suitable using Algorithm 11. The remaining new PQ message D fits in the gap after F_1 and hence, obtains o(D) = 2. The overall priority assignment for the extended message set is shown in Table 4.6 together with the respective WCRTs.

Table4.6: Priority Assignment for the Extended Message Set

Message m	Priority Level $o(m)$	CAN ID id(m)	WCRT R_m
F_1	1	1	7
D	2	2	3
F_2	3	3	4
F_3	4	5	14
F_4	5	7	7
F_5	6	9	7
A	7	10	10
F_6	8	11	14
B	9	12	10
F_7	10	13	18
F_8	11	15	14
F_9	12	17	18
F_{10}	13	19	16
F_{11}	14	21	17
F_{12}	15	23	18
C	16	24	19

$\mathcal{S} := \emptyset$
for *all* $N \in (\mathcal{W} \cap \mathcal{N}^{PQ})$ **do**
    Compute R_m using (4.1) and (4.2) with $f_{k,m} = 0$ for all $k \in \text{hp}(N) \cap \mathcal{M}^{\text{FQ}}$
    if $R_m \leq E_m$ **then**
        $\mathcal{S} := \mathcal{S} \cup \{N\}$
    end
end
if $\mathcal{S} \neq \emptyset$ **then**
    Select $N \in \mathcal{S}$ with largest value of C_N/T_N
    $o(N) := l$; $l := l - 1$; $\mathcal{W} := \mathcal{W} \setminus \{N\}$; $g_{\text{cur}} := g_{\text{cur}} - 1$
    continue with main loop
end
for *all* $G_k \in G^{\text{new}}$ **do**
    Try all $N \in \mathcal{N}^{\text{FQ}} \cap \mathcal{M}_{G_k}$ and find N with the smallest R_N using (4.3) and (4.4)
    if $R_N \leq E_N^{\min}$ **then**
        $\mathcal{S} := \mathcal{S} \cup \{N\}$
    end
end
if $\mathcal{S} \neq \emptyset$ **then**
    Select $N \in \mathcal{S}$ with smallest value of R_N and set `FIFO` = true
    $o(N) := l$; $l := l - 1$; $\mathcal{W} := \mathcal{W} \setminus \{N\}$; $g_{\text{cur}} := g_{\text{cur}} - 1$; $f_N := w_N^{n+1}$
    continue with main loop
end
if $F_n \in \mathcal{M}^{\text{PQ}} \wedge \sum_{i=1}^{n-1} g_i \geq |\mathcal{W}|$ **then**
    Compute R_{F_n} using (4.1) and (4.2) with $f_{k,F_n} = 0$ for all $k \in \text{hp}(F_n) \cap \mathcal{M}^{\text{FQ}}$
    if $R_{F_n} \leq E_{F_n}$ **then**
        $o(F_n) := l$; $l := l - 1$; $\mathcal{W} := \mathcal{W} \setminus \{F_n\}$; $n := n - 1$; $g_{\text{cur}} := g_n$
        continue with main loop
    end
    else
        return not schedulable
    end
end
else if $F_n \in \mathcal{M}^{\text{FQ}} \wedge \sum_{i=1}^{n-1} g_i \geq |\mathcal{W}|$ **then**
    Compute R_{F_n} using (4.3) and (4.4) with $f_{k,F_n} = 0$ for all $k \in \text{hp}(F_n) \cap \mathcal{M}^{\text{FQ}}$
    if $R_{F_n} \leq E_{F_n}^{\min}$ **then**
        $o(F_n) := l$; $l := l - 1$; $\mathcal{W} := \mathcal{W} \setminus \{F_n\}$; $n := n - 1$; $g_{\text{cur}} := g_n$; $f_{F_n} := w_{F_n}$; `FIFO` = true
        continue with main loop
    end
    else
        return not schedulable
    end
end
else
    return not schedulable
end

Algorithm 10: Case 1: No FIFO group spans the current priority-level

4.7 Performance evaluation

In this section, we evaluate the performance of the algorithms proposed in this paper. To this end, Section 4.7.1 compares our improved schedulability algorithm to the algorithm in [25] and Section 4.7.3 demonstrates the applicability of our priority assignment algorithm for the message set extension. Note that our algorithms are implemented in C++ and the computational experiments are run on a laptop computer with i5 core CPU, 2.2 GHz processor, 4 GB RAM.

4.7.1 Performance of the Improved Schedulability Algorithm

In this experiment, the run-time of the existing schedulability algorithm in [25] and our improved schedulability algorithm in Section 4.4.2 are compared. We perform the WCRT analysis for different message sets using our C++ implementation of Algorithm 7 and the implementation of the algorithm in [25]. The message set properties (bus speed, load, number of PQ/FQ messages) and run-time results are summarized in Table 4.7. It is readily observed that the run-time of our improved algorithm is better than that of the original algorithm. In all cases, our algorithm completed the analysis less than the original algorithm. For large message sets such as Set8 (295 messages), improved algorithm is 4 times better than the original algorithm.

Table4.7: Different Message Sets and Their Run-Times

Message Set	Load (%)	Bus Speed (Kbps)	No Of PQ/FQ Mess.	Original Alg. Run-Time [sec]	Improved Alg. Run-Time [sec]
Set 1	32.34	125	55/42	0.009	0.006
Set 2	27.45	125	53/25	0.0044	0.0037
Set 3	31.11	125	65/33	0.009	0.0045
Set 4	34.25	125	68/29	0.009	0.0048
Set 5	25.94	250	97/54	0.022	0.009
Set 6	28.6	250	95/72	0.025	0.011
Set 7	30.1	250	124/44	0.023	0.01
Set 8	24.5	500	134/161	0.08	0.031
Set 9	6.81	500	91/60	0.005	0.003
Set 10	15.9	500	235/90	0.001	0.008

4.7.2 NetCarAnalyzer Comparison

In this experiment, the correctness of the NetCarAnalyzer[7] is discussed. NetCarAnalyzer uses existing schedulability algorithm in [25], but we show that in Figure 4.3, it has some flaws. Although it is a commercial product, we easily found its mistakes. We compared the NetCarAnalyzer results and original algorithm that we have written in C++. First, it is observed that NetCarAnalyzer does not support FIFO-symmetry, which is a key concept in the algorithm, messages in the same FIFO node have different response times. When we look at the upper parts of the red line in Figure 4.3, we see that there are three levels, one for each FIFO node. Messages of first FIFO node has WCRT value of 69, second FIFO node has WCRT value of 72 and the third one has 78. But blue line does have such levels for FIFO nodes. Secondly, NetCarAnalyzer gives wrong WCRT's such that WCRT values are lower than we calculated. For the message with priority 12, NetCarAnalyzer result is 28 ms. This is a FIFO message and the lowest priority message in the same FIFO node has priority 64, it roughly means that there are 63 messages before that message wins arbitration, therefore WCRT of messages at that node should be higher than 63 ms(each message transmission is approximately 1 ms).

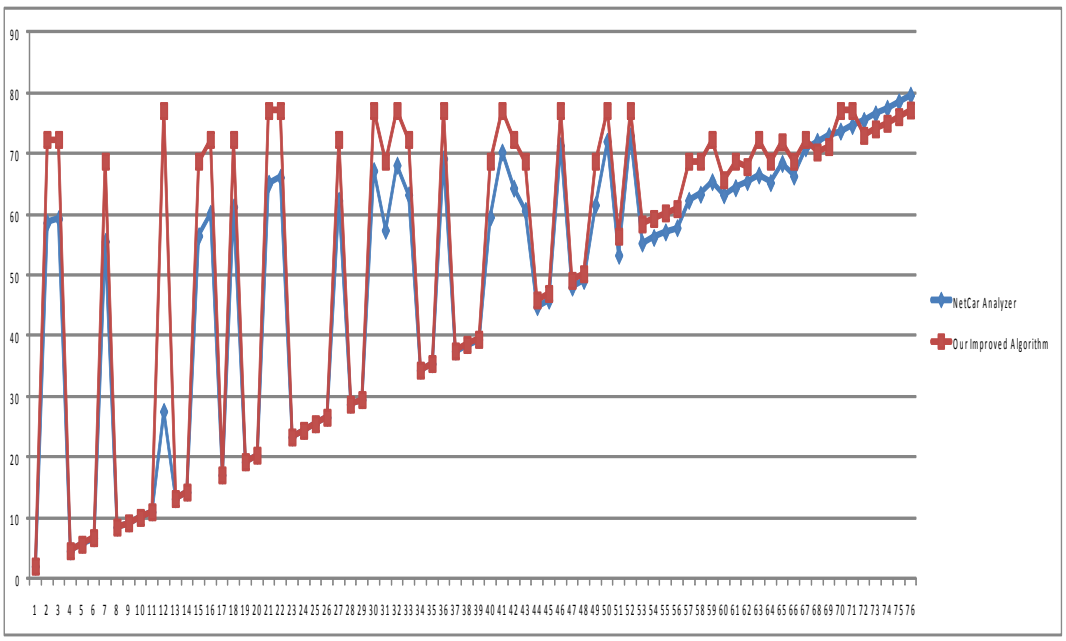

Figure 4.3: NetCarAnalyzer Results and Our Implementation Results

4.7.3 Priority Assignment for Message Set Extension

In order to show the efficiency of the message set extension algorithm, we extend the message sets in Table 4.7. For each message set, we generate a smaller message set to be added to the CAN network with the existing message set. Some of the new messages are PQ and some of them are FQ messages. Our experiment proceeds as follows. First, we determine the smallest bus speed for each message set such that Algorithm 9 obtains a feasible priority assignment. Then, at the same bus speed, we perform a manual priority assignment by placing the new messages in available gaps in a deadline monotonic way. Hereby, the manual assignment may give schedulable or unschedulable results. Hence, we record if the manual assignment leads to a schedulable priority assignment as well as how many messages violate their deadlines. Table 4.8 summarizes our results. In most cases, the manual assignment cannot determine a feasible priority assignment, whereby up to 30% of the new messages can be infeasible.

Table4.8: Extension Sets and Priority Assignment Results

Message Set	Minimum Speed Required For Improved Alg.	PQ Mess.	FIFO Mess.	Manual Assignment Result with the Same Bus Speed
Extension Set 1	232 Kbps	10	8	Not schedulable(1 unsch. mess.)
Extension Set 2	208 Kbps	6	13	Not schedulable(2 unsch. mess.)
Extension Set 3	128 Kbps	11	8	Not schedulable(11 unsch. mess.)
Extension Set 4	268 Kbps	2	11	Schedulable
Extension Set 5	392 Kbps	13	20	Not schedulable(1 unsch. mess.)
Extension Set 6	445 Kbps	14	14	Not schedulable(3 unsch. mess.)
Extension Set 7	468 Kbps	6	17	Not schedulable(2 unsch. mess.)
Extension Set 8	715 Kbps	45	15	Not schedulable(3 unsch. mess.)
Extension Set 9	245 Kbps	11	25	Not schedulable(2 unsch. mess.)
Extension Set 10	484 Kbps	23	31	Not schedulable(3 unsch. mess.)

CHAPTER 5

GATEWAY ANALYSIS

5.1 Overview

This chapter is about analyzing and scheduling CAN networks which contains gateways. With the increasing number of electronic units in cars, gateways become very common in CAN systems. They are needed because of several reasons. Gateways provide message transfer between two isolated CAN networks. The network inside a single car may need to be disjointed and several separated networks form. Even though the two networks are isolated, they still need to communicate with each other. The main contributions of this chapter are as follows: First, in Section 5.2, we explain the necessity of gateways. Section 5.3 introduces some commercial gateway applications and describe their main features. Then, in order to guarantee the real-time capability of systems with gateways, we make schedulability analysis in Section 5.4. This analysis is very different from traditional analysis and requires additional concepts. In the literature, gateway schedulability analysis exists, we have used [45] paper in this section. However, in the literature there is no study about message scheduling in gateway systems. In Section 5.5, we have developed a scheduling algorithm for CAN networks containing gateways. This is the one of the major contributions of this chapter. Another major contribution is about FIFO queues. So far, gateway and FIFO concepts have not been studied together in the literature. In Chapter 4, we have examined FIFO CAN controllers. Using the experience obtained in FIFO, we have developed a schedulability analysis solution for systems consisting of PQ, FIFO controllers and gateways. Moreover, in our analysis, gateways can be either FIFO or PQ. In Section 5.6, this solution is introduced. Finally, in Section 5.7, we verify our

algorithms with case studies.

5.2 Why Gateways are Needed

The reasons for the usage of gateways are numerous and these reasons are crucial. This is why car manufacturing companies such as PSG, BMW are using them in CAN systems ([32]). One of the reasons is the reduction of complexity. In a large CAN network, not all ECU's communicate with each other. For example, multimedia components do not need to communicate with body part components inside a car. Multimedia messages only travel among ECU's related to multimedia. Therefore, we can divide a large network into several isolated networks in which unrelated parts stay separated. Similarly, a car can be divided into domains based on functionality of parts.

Car manufactures divide a car into domains as: powertrain, chassis, body, safety and telematics ([34]). These domains have different performance, safety and communication requirements. Powertrain domain is responsible for power generation, engine control and transmission. It needs communication in miliseconds periods. Chassis is responsible for suspension, steering and braking, and its communication messages have hard deadlines. Another domain is safety. Airbags, cruise control and tire pressure monitoring are some example components of that domain. These examples need high speed data transmission. Body domain mainly implements body and comfort functions. Applications in this domain are not safety critical and do not require high bandwidth. Communication mostly depends on driver or passengers' action. Telematics domain includes multimedia, navigation and entertainment systems. In this domain, huge amount of data is required to be exchanged. Since these domains are irrelevant to each other, they are separated. They have different communication needs, and their communication networks can be separated as well. However, it may not be possible to fully isolate these networks. Some messages called *inter-domain* messages are required among networks. These messages need to travel in all networks and gateways should allow them. Note that inter-domain messages are not as frequent as another kind of messages called *intra-domain* messages. They are the messages which travel only in their own network. If they had the same frequency, there would

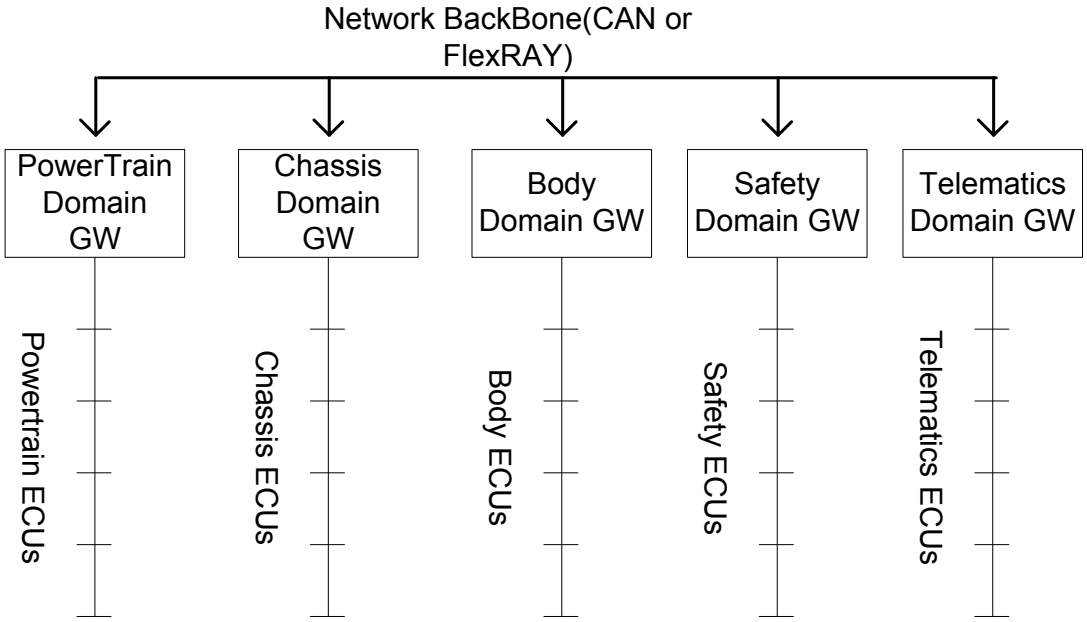

Figure 5.1: Car domains

Table5.1: Properties of BMW 7 series domains

	Powertrain	Chassis	Body	Telematics	Safety
No of ECUs	3-6	6-10	14-30	4-12	11-12
Bandwidth	500 Kbps	500 Kbps	100 Kbps	22 Mbps	10 Mbps
No of Messages	36	180	300	660	20
Cycle Time	10 ms-10 s	10 ms-10 s	50 ms-2 s	20 ms-5 s	50 ms
Safety Requirements	high	high	low	low	very high

be only one network, which is the reason of dividing networks. Overall, network of a car is separated into domains to reduce the complexity and different domains are connected together and this is done via gateways.

BMW 7 series is equipped with highly sophisticated electronic components. Table 5.1 shows features of each domain in BMW 7 series ([37]). As seen from table, bandwidth, cycle time and safety requirements of each domain are different. Body domain has 300 messages and 100 Kbps bus speed is adequate. However, safety domain has 20 messages and it needs 10 Mbps bus speed. Then, it was wise to divide into parts. If they did not divide them, either they would use 10 Mbps speed for all networks and increase cost and complexity or use slower speeds and allow messages violating their deadlines.

Second reason why we need gateways is about bus speed and cable length. Shorter bus length means higher bus speed. With 1 Mbps bus speed, we can have at most 40 m cable and with 100 Kbps speed, we can have 500 m cable. By using gateways, we are adding networks end-to-end and hence shortening the bus length. This is a way of increasing bus speed. Similarly, we can increase the communication network distance.

The final reason is due to reliability and safety needs. Domains with different reliability and safety needs are separated. If an error occurs in a domain, it affects only that domain, the other domains keep safe. When CAN bus system on one side of the gateway happens some error, the system on the other side can still work correctly.

5.3 Commercial CAN2CAN Gateways

In order to make a schedulability analysis for gateways, we need to know certain characteristics of them. We have made a deep research on gateways, found that there are two kinds of gateways (when we talk about CAN to CAN gateways, it is also called bridge): Simple gateways and advanced gateways. Before describing their features, we should define some gateway related concepts such as message filtering, ID translation, extra buffering time, signal extraction and integration.

CAN bus works in broadcast. All messages sent from any node arrive to all nodes. If the arrived node is interested in that message, it takes it, otherwise, simply rejects. In gateways also this is an important property. All messages arrive to the gateway from two networks. Gateway has two ports, port 1 for network 1 and port 2 for network 2. All messages traveling in network 1 arrive to gateway port 1 and all messages traveling in network 2 arrive gateway port 2, but gateway only transfers inter-network messages. Intra-domain messages are rejected. Both ports filter irrelevant messages, this feature is called message filtering.

In the two networks, same message can have different priorities. It may have a lower priority in one network and may have a higher priority in the other, since the two networks are actually isolated. During the message passing, its ID can be changed. This is called ID translation. A gateway with extra buffering time stores messages before

delivering to the other network. This feature provides some advantages like reduced jitter. Finally, signal extraction and integration in the gateways can be possible. In CAN messages, several signals are packed. The goal of signal packing for CAN is packing different signals in the same message in order to reduce the framing overhead and hence improve the schedulability of CAN systems. In the gateway, a message can be broken up into signals and then different signals from different messages can be packed together. These are the features that need attention during WCRT analysis.

5.3.1 Simple Gateways:

They are two port simple devices, have message filtering and ID translation properties. But no signal extraction or message integration feature exists. Also they have no additional wait support in the buffers. There is one output buffer for each port, incoming messages are checked and mapped from bridge map and put to the output buffer. In the output buffers, they enter into arbitration.

We have checked the datasheets of several different products and they all have these properties in common. They are configured via RS232. This configuration only includes CAN bus speed adjustment and ID mapping. In the configuration programs for these products, only we set IDs and mappings for messages. The available simple gateways from the market are PCAM Router([5]), I-7532([2]), SW-400T([1]), CG-ARM7([4]), Ixxat([3]).

5.3.2 Advanced Gateways:

Advanced gateways are implemented in FPGAs and they may have more than two ports. They have message filtering,ID translation, signal extraction, message integration properties. Also timeout events can be triggered.(Extra buffering is possible). But there is not much information on these kinds of bridges since they are network-dependent special designs ([27, 28]).

In this chapter, the analysis suggested is about simple gateways, since they are more common. We have considered the features of them while developing the ideas and

algorithms.

5.4 Schedulability Analysis

In this section, we study the analysis described in [45] paper. We calculate WCRT of all messages inside a two-network system connected via a gateway. Adding a GW to the network brings extra challenges. One of the challenges is that GW messages obtain additional jitters. When we want to transfer a message in a traditional CAN bus, the response time of that message cannot be calculated constantly. This response time value is between best case response time(BCRT) and WCRT. There is a variability in it. So far we have been only interested in WCRT. Now, consider that we transfer a message from source network to the destination network via GW. Response time of that message in the source network cannot be recorded as constant as well. The variability in the response time of the message on the source network typically translates into queuing jitter on the destination network. This means that the arrivals of that message to the GW is not periodic. For example, it is generated in the source network in $10msec$ period. The instances of the message are generated at $t = 0, 10, 20...$ That instances may arrive to the GW at $t = 2, 18, 22$. Then, the instances inherit jitter while passing through the GW. In the worst-case, multiple instances of the same message that were originally queued periodically on the source network, may end up being transmitted back-to-back on a destination network. In this case, lower priority messages may undergo increased delays.

The problem does not end here. There is also a mutual dependence issue: The release time of a message in network 1 depends on the GW messages coming from network 2. Similarly, release time of a message in network 2 depends on the GW messages coming from network 1. They affect each other and we cannot calculate WCRT of network 1 messages without knowing the WCRT's of network 2 messages or vice versa.

As a result, the scheduling analysis problem of systems with GW's cannot be trivially solved because of these two problems. There is a concept in distributed real time analysis called *holistic response time analysis*. Holistic means 'whole'. There are

multiple processors in distributed systems and some information is exchanged among these processors, tasks communicate by message passing which is similar to bus and GW ([42]). CAN counterpart of holistic analysis of distributed real time systems is CAN GW analysis. We have searched literature about both holistic analysis and CAN GW analysis. The work in [45], uses the same holistic approach developed for distributed real time systems and proposes a scheduling analysis solution for CAN networks with GW's. We have studied and implemented this solution.

Holistic GW analysis can be used to determine the overall end-to-end response time of each message, and thus determine if end-to-end deadlines are met. It assumes that messages inherit all of the response time up to their reception at a GW plus the maximum delay in being processed by the GW as queuing jitter on the destination network.

The mutual dependence issue is also solved. The solution consists of analyzing the messages of two networks simultaneously in decreasing priority order. By doing so, we can know the jitters of high priority messages when working with a low priority message. Same priority non-GW messages may exist in both networks, but GW messages have to have unique priorities in both networks.

For the following parts, we assume that system consists of two networks which are connected via a simple CAN2CAN gateway: Network 1 and Network 2. Two networks may have same or different bus speeds. In each network, there are CAN controllers connected to their own buses. Messages could be gateway(GW) or non-gateway(non-GW) messages. For GW messages, the generating network is called source network and target network is called destination network.

Table5.2: Notation For Message m

Notation For Message m			
R_s	Source network response time	$dlp(m)$	Lower priority messages than m in the destination network
R_d	Destination network response time	$slp(m)$	Lower priority messages than m in the source network
R_{e2e}	End-to-end total response time	$shp(m)$	Higher priority messages than m in the source network
w_s	Source network buffering delay	$dhp(m)$	Higher priority messages than m in the destination network
w_d	Destination network buffering delay	$shp_{gw}(m)$	Higher priority gw messages from source network
B_s	Blocking due to source network	$dhp_{gw}(m)$	Higher priority gw messages from destination network
B_d	Blocking due to destination network	ADC	Absolute distance constraint
$C_{gwmin12}$	Minimum transmission time of FIFO gw messages from network 1 to network 2		
$C_{gwmax12}$	Maximum transmission time of FIFO gw messages from network 1 to network 2		
$C_{gwsum12}$	Total transmission time of FIFO gw messages from network 1 to network 2		

With WCRT analysis, we mean end-to-end delay analysis, R_{e2e} is the total response time((5.1)). WCRT analysis for both non-GW and GW messages are so different from traditional analysis. For GW messages, response time is the time between generation in the source network and reception in the destination network, it is the sum of response time in the source network R_s and response time in the destination network R_d . For non-GW messages, it is simply the time between generation time and reception time in the same network. While calculating WCRT's, we treat GW and non-GW messages differently. A GW has two queues, one queue for messages from network 1 to network 2 and another for messages from network 2 to network 1. For example, $C_{gwmin12}$ is for the queue from network 1 to network 2 and $C_{gwmin21}$ is for from network 2 to network 1.

$$R_{e2e} = R_s + R_d \tag{5.1}$$

While calculating WCRT, different types of messages cause blocking and interference. B_s is due to blocking message in the source network and B_d is in the destination network. $shp(m)$ and $slp(m)$ represent two sets of messages coming from the source network and having higher and lower priorities. $dhp(m)$ and $dlp(m)$ are the other two sets of messages from destination network. Higher priority GW messages from source and destination networks are $shp_{gw}(m)$ and $dhp_{gw}(m)$, respectively.

We used and implemented Algorithm 13 for WCRT analysis. It starts with sorting the

messages of two networks in decreasing priority order. For each message, we repeat the same steps. For the worst case, at critical instant, all higher priority messages are assumed to be released. GW messages are released with jitter and these jitter values are previously calculated since we analyze in decreasing priority order. This critical instant idea puts some pessimism into the analysis. All GW messages cannot be released at the same time since their reception from the other network is done one by one. We can consider arrival of GW messages as if they were released with offset. Then the release order of GW messages affects the response time. As we mentioned, jitter for GW messages is the difference between WCRT and BCRT. In the analysis, since we do not know the actual arrival order messages into the GW, there are several possibilities for BCRT. The second for loop shows this. For each BCRT possibility, the analysis is repeated and largest of the response time values are chosen as WCRT. We use Absolute Distant Constraint(ADC) term for different arrival order possibilities. We use the ADC_k to indicate the time distance between m_k and the first arrived $dhp_{gw}(i)$ messages. ADC_k of the messages not only depend on their own C_k, but also on the arriving order of the messages. (5.2) shows how to calculate ADC and for the first arrived message $ADC_k = 0$.

$$ADC_k = C_k + ADC_j \;\; where \;\; level(m_j) = level(m_k) - 1 \tag{5.2}$$

$level(m_j) = level(m_k) - 1$ means the arrival order, m_k is the next message after m_j. Assume m_1, m_2 and m_3 are messages with $C_{m1} = 1, C_{m2} = 2, C_{m3} = 2$. If the arrival order is m_1, m_2, m_3, then $ADC_1 = 0, ADC_2 = 2, ADC_3 = 4$. If the arrival order is m_2, m_1, m_3, then $ADC_2 = 0, ADC_1 = 1, ADC_3 = 3$. In the analysis, taking ADC into account brings extra overhead and run-time of the algorithm increases. We can reduce algorithm complexity if we exclude ADC, this gives approximate result rather than exact but the difference is not major. In the case studies, we will use approximate case.

5.4.1 Non-GW Messages

If the analyzed message is a non-GW message, analysis is similar to traditional analysis. There are two sources of delay: blocking delay and interference delay. Blocking

input : Message set $\mathcal{M}$ with priority assignment $o(m)$ for each message $m \in \mathcal{M}$

output: WCRT R_m for each message $m \in \mathcal{M}$

Initialize $R_{m,max} = 0$ for all $m \in \mathcal{M}$
for *Each priority l, highest priority first* **do**
    for *Each different possible arrival order of GW messages* **do**
        Let m be the message such that $o(m) = l$
        Calculate ADC of each message for each arrival order of GW messages
        if *m is not a gateway message %% We only look at the source network* **then**
            Compute $R_{s,m}$ using Algorithm 14
            $R_m = R_{s,m}$
        end
        else if *m is a gateway message* **then**
            Compute $R_{s,m}$ and $R_{d,m}$ using Algorithm 14
            $R_m = R_{s,m} + R_{d,m}$
        end
        if $R_m > R_{m,max}$ **then**
            $R_{m,max} = R_m$
        end
    end
end

Algorithm 13: WCRT analysis for CAN-CAN gateways.

delay B_s is due to lower priority messages in the same network or due to lower priority GW messages. We use (5.3) for blocking delay.

$$\begin{aligned} B_{s,i} = max(C_i, C_m, C_l) \\ m \epsilon\ slp(i) \\ l \epsilon\ dlp_{gw}(i)) \end{aligned} \tag{5.3}$$

Interference delay is due to higher priority messages from either same network or from the other network. For the GW messages coming from the other network, we have to include their jitters as seen from (5.4). Initial value for the iteration is $w^0_{s,i} = C_i$.

Here, we subtract ADC_k because message is released from GW at $t = ADC_k$, not at $t = 0$.

$$w_s^{n+1} = B_s + \sum_{shp}(m)\left\lceil w_s/T_j \right\rceil C_j + \sum_{dhp_{gw}(m)} \left\lceil \frac{w_s + R_s - C_k - ADC_k}{T_k} \right\rceil C_k \tag{5.4}$$

For the response time, we take the largest of the buffering delay:

$$R_{s,i} = max(w_{s,i}^n) + C_i \tag{5.5}$$

input : Message m, message set $\mathcal{M}$ with priority assignment $o(m)$ for each message $m \in \mathcal{M}$

output: WCRT $R_{s,m}$

Initialize $w_{s,m}^0 = B_m$, $n = 0$
if *m is not a gateway message %% We only look at the source network* **then**
  while $w_{s,m}^{n+1} \neq w_{s,m}^n$ **do**
    for *Messages* $k \in hp(m)$ **do**
      if $k \notin dhp_{GW}(m)$ *%% message on source network* **then**
        $w_{s,m}^{n+1} = w_{s,m}^{n+1} + \lceil \frac{w_{s,m}^n}{T_k} \rceil C_k$
      end
      else
        $w_{s,m}^{n+1} = w_{s,m}^{n+1} + \lceil \frac{w_{s,m}^n + R_{s,k} - C_m - ADC_k}{T_k} \rceil$
      end
    end
  end
  $R_{s,m} = w_{s,m}^{n+1} + C_m$
end

Algorithm 14: WCRT Computation for non-GW messages

5.4.2 GW Messages

If the analyzed message is a GW message, analysis consists of two parts: source network delay and destination network delay. Analysis for the source network delay is same as analysis for non-GW messages. We ignore processing delay inside the GW.

The waiting time in the GW during arbitration in the destination network is included in destination network delay. Note that different from source network, blocking delay is only due to lower priority messages in the destination network((5.6)). (5.7) and (5.8) are also similar to source network calculations.

$$B_{d,i} = max(C_i, C_m) \quad m \in dlp(i) \tag{5.6}$$

$$w_d^{n+1} = B_d + \sum_{dhp(m)} \left\lceil w_d / T_j \right\rceil C_j + \sum_{shp_{gw}(m)} \left\lceil \frac{w_d + R_s - C_k - ADC_k}{T_k} \right\rceil C_k \tag{5.7}$$

$$R_{d,i} = max(w_{d,i}^n) + C_i \tag{5.8}$$

input : Message m, message set $\mathcal{M}$ with priority assignment $o(m)$ for each message $m \in \mathcal{M}$

output: WCRT $R_{s,m}$ and $R_{d,m}$

Initialize $w^0_{s,m} = B_{s,m}$, $w^0_{d,m} = B_{d,m}$,
while $w^{n+1}_{s,m} \neq w^n_{s,m}$ %% *Source Network* **do**
  for $k \in hp(m)$ **do**
    if $k \notin dhp_{GW}(m)$ **then**
      $w^{n+1}_{s,m} = w^{n+1}_{s,m} + \lceil \frac{w^n_{s,m}}{T_k} \rceil C_k$
    end
    else
      $w^{n+1}_{s,m} = w^{n+1}_{s,m} + \lceil \frac{w^n_{s,m} + R_k - C_m - ADC_k}{T_k} \rceil C_k$
    end
  end
end
$R_{s,m} = w^{n+1}_{s,m} + C_m$
while $w^{n+1}_{d,m} \neq w^n_{d,m}$ %% *Destination Network* **do**
  for $k \in hp(m)$ **do**
    if $k \notin shp_{GW}(m)$ **then**
      $w^{n+1}_{d,m} = w^{n+1}_{d,m} + \lceil \frac{w^n_{d,m}}{T_k} \rceil C_k$
    end
    else
      $w^{n+1}_{d,m} = w^{n+1}_{d,m} + \lceil \frac{w^n_{d,m} + R_k - C_m - ADC_k}{T_k} \rceil C_k$
    end
  end
end
$R_{d,m} = w^{n+1}_{d,m} + C_m$

Algorithm 15: WCRT computation for GW messages

5.5 Scheduling Algorithm

Assigning priorities to several networks at the same time may bring many complications and it is hard without a systematic approach. Existing scheduling algorithms do not support two networks and a GW. Moreover, as mentioned in the previous section GW messages have an unpredictable jitter issue. This problem has not been studied before and we developed an algorithm which assigns priorities to messages of two networks connected via GW.

One can try to schedule two networks separately with known priority assignment methods. However, even if the two networks are separately schedulable, overall it may be unschedulable. For instance, a GW message may be given low priority in one of the networks and it easily violates its deadline. This is the simplest case and because of such cases, we developed a scheduling algorithm for GW's.

The algorithm starts with an assumption for GW messages. Since we do not know the jitters before assigning any priority, we assume that the jitters of GW messages are equal to their periods to capture the worst cases. This is a very pessimistic assumption and increases the effect of GW's to the system. For all GW messages initially:

$$J_m = T_m \tag{5.9}$$

We keep two priority tables for two networks, at the end the GW messages have the same values on both tables, meaning that GW messages have unique priorities on both networks. In Algorithm 16, we present our ideas. We start to assign priorities with network 1 non-GW messages and we match the lowest priority levels to those messages first. If a message is schedulable at this priority level, we place it there and this continues until no more new messages are schedulable. Then, we start non-GW messages of network 2 and check whether they are schedulable starting from the lowest priority level. When no more non-GW messages are schedulable, we pass to GW messages and try them one by one. For a priority level, we try all unassigned GW messages, we run WCRT analysis assuming unassigned messages are at higher priorities. Then, we choose the best suitable message.

In the algorithm, after all messages are visited once, we find a priority level for each message and then adjust the message priorities to have the same priority for GW messages. Moreover, we place back the original jitters of GW messages. We make WCRT analysis for the assigned priority levels. If the system is schedulable, we terminate the algorithm and if not schedulable, we try to make unschedulable messages by swapping process.

In this step, there is no possibility that non-GW messages are unschedulable. Because in the previous steps, if a non-GW message is unschedulable, we have to terminate the algorithm. Even if the GW message is unschedulable, we assign a priority to it. Therefore, here we may have unschedulable GW messages. We can make them schedulable by swapping them with higher priority schedulable non-GW messages.

Before swapping:	non-GW message:	WCRT = 67 ms	Deadline = 100 ms
	GW message:	WCRT= 107 ms	Deadline = 100 ms
After swapping:	GW message:	WCRT = 94 ms	Deadline = 100 ms
	non-GW message:	WCRT= 75 ms	Deadline = 100 ms

Note that this swapping should be between GW messages and non-GW messages. It cannot be between two GW messages because changing the order of them does not make them schedulable if at least one of them is unschedulable. However, swapping unschedulable GW message with a schedulable non-GW message may also result in deadline violation. In that case, all higher priority non-GW messages are tried and if none of them is schedulable, we terminate.

input : Message set $\mathcal{M}$

output: Priority order o if schedulable

Initialize For all gateway messages: $R_{s,m} = D_m/2$, $R_{d,m} = D_m/2$, for all other messages: $R_m = D_m/2$, $\mathcal{W} = \mathcal{M}$, $l := |\mathcal{W}|$, $J_m = T_m$

while *Priority assignment is not schedulable* **do**

  while $\mathcal{W} \neq \emptyset$ **do**

    for *Non-GW messages on network 1* **do**

      if *Message m is schedulable at level l* **then**

        $o(m) = l$ and $\mathcal{W} = \mathcal{W} \setminus \{m\}$ and $l := l - 1$

      end

    end

    for *Non-GW messages on network 2* **do**

      if *Message m is schedulable at level l* **then**

        $o(m) = l$ and $\mathcal{W} = \mathcal{W} \setminus \{m\}$ and $l := l - 1$

      end

    end

    for *GW messages m in* $\mathcal{W}$ **do**

      Compute $R_{m,s}$ and $R_{m,d}$ using the equations in subsection 5.4.2

      Define $\Delta_m = D_m - R_{m,s} - R_{m,d}$

    end

    Determine m with the smallest Δ_m

    $o(m) = l$ and $\mathcal{W} = \mathcal{W} \setminus \{m\}$ and $l := l - 1$

  end

  $J_m = J_m^{original}$

  Use Algorithm 13 to compute R_m for each $m \in \mathcal{M}$

  if $R_m \leq D_m$ *for all* $m \in \mathcal{M}$ **then**

    Modify priorities of each network such that GW messages have same ID's on both sides

    return o

  end

  else

    Swap GW messages with $R_m > D_m$ with neighboring non-GW messages until all messages are schedulable

    if *Swapping does not achieve a schedulable priority order* **then**

      return not schedulable

    end

  end

end

Algorithm 16: Scheduling for CAN gateway networks.

5.6 Gateways and FIFO Queues

In this section, we further continue on FIFO analysis. In the FIFO chapter, we investigated networks with both FIFO and priority queued CAN controllers. Now, we see what happens if we add a GW to this system. This GW can be either FIFO or priority queued gw. In the [25] paper, it is stated that it is very common to use FIFO queues inside GW's because number of GW messages can be larger than the number of available buffers in a priority queue. Therefore using FIFO queues in GW's is a solution. Although paper contains experiments with FIFO queued GW's, it does not give any analysis method. As far as we know, there is no published WCRT analysis on GW's and FIFO queues together. The other controllers also can be either FIFO or priority queued. For calculating WCRT's of messages, we combine FIFO analysis and GW analysis.

In the FIFO analysis, we have seen that a FIFO queued message can be modeled as a priority queued message but with additional jitter. Total waiting time in the FIFO controller is considered as buffering jitter f_k. If the GW is also FIFO, jitters of messages become larger and algorithm becomes more complicated in this case. The difference between FIFO GW analysis and PQ GW analysis is that in PQ case jitter of GW messages is R_s, but if the GW is FIFO, jitter becomes $R_s + f_k$. For GW case FIFO buffering is modeled with R_d. In FIFO GW buffer, GW messages wait $R_d - C_m$ time before entering into arbitration. Then total jitter becomes $R_s + R_d - C_m$.

Note that we have a *while* loop. As we calculate from higher to lower priority messages, f_k values are needed to be updated. We cannot use the improved schedulability analysis described in FIFO chapter. Because here in order to break the mutual dependence, we have to go in decreasing order, but improved algorithm works in increasing priority order.

5.6.1 Non-GW Messages:

For non-GW messages, we have only source network calculations. The equations are similar to Section 5.4 equations. We have $(C_{sum} - C_{min})$ for FIFO buffers, f_m and $(R_d - C_m)$ different from there. Algorithm 13 is written for the case where GW has

input : Get $C_{min}, C_{max}, C_{sum}$ for FIFO nodes and $C_{gwmin}, C_{gwmax}, C_{gwsum}$

```
1  repeat = True
2  while repeat do
3      for Each message m, highest priority first do
4          //Source Network Calculations
5          if m is a PQ message in the originating node then
6              Use (5.10),(5.12) and (5.13)
7          end
8          else if m is a FIFO message in the originating node then
9              Use (5.11),(5.12) and (5.13)
10             if f_m ! = w_s then
11                 f_m = w_s
12                 repeat = True
13             end
14         end
15         // Destination Network Calculations
16         if m is a GW message then
17             Use (5.14),(5.15),(5.16) and 5.17
18         end
19     end
20 end
```

Algorithm 17: Gateway Schedulability Analysis For Mixed Systems.

FIFO buffer, if it has PQ buffer, then it needs slight modification. For PQ GW case R_d is dropped from (5.12). If the originating node of non-GW message is FIFO, we have to take $C_{sum}, C_{max}, C_{min}$ values into consideration.

$$\begin{aligned} B_{s,i} = max(C_i, C_m, C_l)\ for\ PQ\ message \\ m \epsilon\ slp(i) \\ l \epsilon\ dlp_{gw}(i)) \end{aligned} \tag{5.10}$$

$$B_{s,i} = C_{sum} - C_{min}\ for\ FIFO\ message \tag{5.11}$$

$$w_s^{n+1} = B_s + \sum_{shp} \left\lceil (w_s + f_j)/T_j \right\rceil C_j + \sum_{dhp_{gw}} \left\lceil \frac{w_s + R_s + R_d - C_k}{T_k} \right\rceil C_k \tag{5.12}$$

$$R_{s,i} = w_{s,i} + C_i \tag{5.13}$$

5.6.2 GW Messages:

For GW messages source network calculations do not change, same with the non-GW messages. For the destination network calculations, if the GW is FIFO, we calculate only contributions due to different network messages, contributions of the same network messages are already included in C_{sum} value. We use (5.15) in this case. If the GW is PQ, we take contributions of two networks into account and use (5.16)

$$\begin{aligned} B_{d,i} = max(C_i, C_m) \\ m \epsilon\ dlp(i) \end{aligned} \tag{5.14}$$

$$w_d^{n+1} = B_d + \sum_{dhp} \left\lceil (w_d + f_m)/T_j \right\rceil C_j \tag{5.15}$$

$$w_d^{n+1} = B_d + \sum_{dhp} \left\lceil (w_d + f_m)/T_j \right\rceil C_j + \sum_{shp_{gw}} \left\lceil \frac{w_d + R_s - C_k}{T_k} \right\rceil C_k \tag{5.16}$$

$$R_{d,i} = max(w_{d,i}^n) + C_i \tag{5.17}$$

5.7 Case Study

5.7.1 Case Study 1

In this experiment, the reliability of GW schedulability analysis described in 13 is examined. For this study, we used a system of two networks and five different message sets. The properties of these sets are given in Table 5.3, number of messages and load levels are different for two networks. Message periods are 20, 50, 100, 500, 1000, 2000 ms message payloads are between 1 and 8 bytes and message priorities are assigned in deadline monotonic order. All nodes have PQ buffers. For each of the message sets, we conducted two simulations using RtaW-Sim: 6-hour simulation and 7-day simulation. 7- day simulations ended between 40 minutes and 70 minutes. Then we ran our algorithm for the same message sets. WCRT Analysis For Message Sets figures compare simulations and algorithm results.

Table5.3: Message Set Groups For Gateway Analysis

	Network 1				Network 2			
	GW Mess.	NGW Mess.	Load	Bus Speed	GW Mess.	NGW Mess.	Load	Bus Speed
Set 1	6	61	23 %	125 Kbps	5	16	23.5 %	125 Kbps
Set 2	14	128	43 %	250 Kbps	19	67	46.5 %	125 Kbps
Set 3	13	202	50 %	250 Kbps	12	183	42.3 %	250 Kbps
Set 4	17	87	67.5 %	125 Kbps	12	99	64.1 %	125 Kbps
Set 5	14	103	48 %	500 Kbps	11	259	80.4 %	250 Kbps

It is seen from all the figures that simulation results never exceeds our results and hence our algorithm is safe. There is difference between 6-hour simulation and 7-day simulation, 7-day gives worse results. Moreover, we notice the jumps and drops in the WCRT graphs. This is due to the fact that we combine the results of the messages of two networks in a single plot. It is clear from WCRT analysis for message set 2

figure that there are two levels in the results. For the algorithm result, the lower level starts from 0 and goes to 100. The upper level starts from 0 and goes to 230. The first level is the result of network 1 which has 250 Kbps speed and the second level shows the results of network 2 which has 125 Kbps speed. The WCRT's of two networks are not close to each other in this example due to different bus speeds.

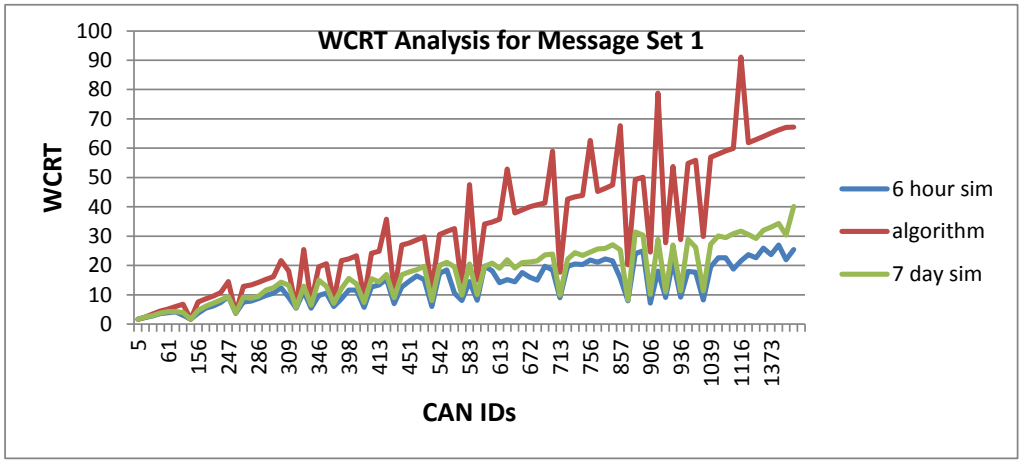

Figure 5.2: Case Study 1: WCRT analysis for message set 1

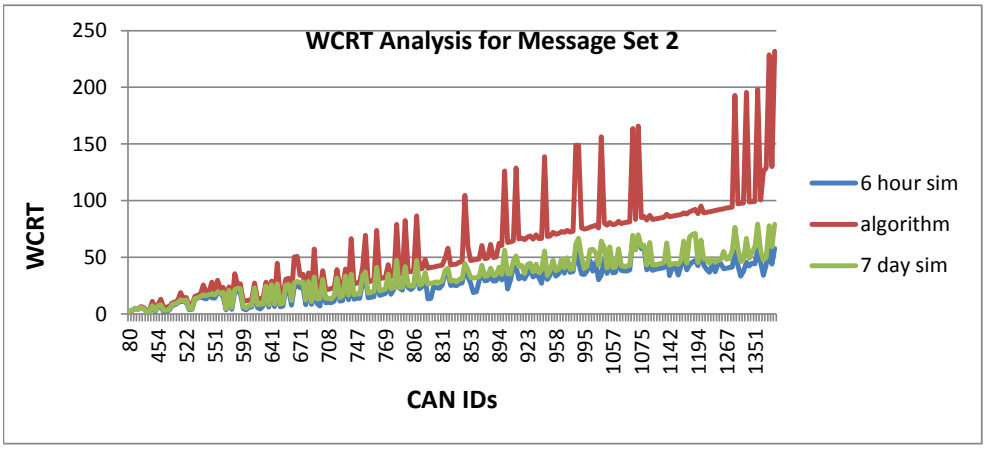

Figure 5.3: Case Study 1: WCRT analysis for message set 2

The gateway messages also have higher WCRT values since they travel in both networks. The load levels of the two networks in message set 1 is similar and the

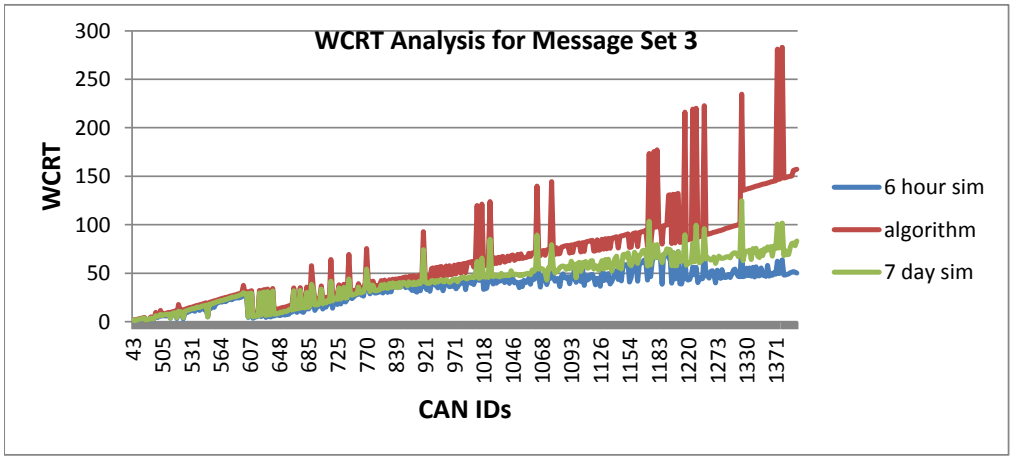

Figure 5.4: Case Study 1: WCRT analysis for message set 3

WCRT's of gateway messages can be better seen. There are in total of 11 gateway messages in message set 1 and there are 11 jumps in the WCRT analysis for message set 1 figure.

The load levels of the two networks in message set 4 are similar and the bus speeds are same, so the difference between the first level and second levels in the WCRT analysis for message set 4 figure is trivial. The jumps and drops are mostly due to gateway messages.

WCRT analysis for message set 3 and 5 figures have similar characteristics with the other figures, but since the total number of messages in message set 3 and message set 5 are high, the jumps and drops are more frequent.

5.7.2 Case Study 2

In order to show the efficiency of GW scheduling algorithm developed in 16, we conducted a comparison experiment. We used the message sets in Case Study 1. First, the messages had unassigned priorities, we assigned priorities using Algorithm 16, also we found minimum bus speed required to make the system schedulable at these priority levels. Since in some of the message sets, the speeds of two networks

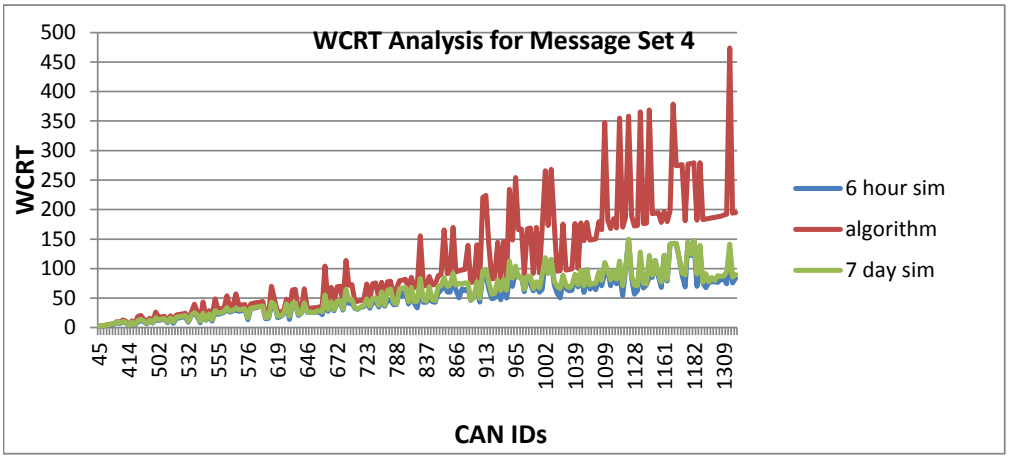

Figure 5.5: Case Study 1: WCRT analysis for message set 4

are different, we decreased two speeds at the same rate to find the minimum speeds required. Then, at these speeds, we scheduled two networks separately and found the number of unschedulable messages. In Table 5.4, for the found bus speeds, the number of unschedulable messages can be seen. Moreover, we measured how many times the swapping part worked in Algorithm 16.

Table5.4: Minimum Speeds Required For Gateway Scheduling and Number Of Unschedulable Messages if networks are separately scheduled or gateway scheduled

Set No	Bus Speed 1	Bus Speed2	Unsch. Mess. two NWs separately scheduled	Unsch. Mess. GW scheduled	No Of Swaps
Set 1	35 Kbps	35 Kbps	15	0	0
Set 2	196 Kbps	71 Kbps	31	0	7
Set 3	134 Kbps	134 Kbps	13	0	0
Set 4	99 Kbps	99 Kbps	34	0	40
Set 5	476 Kbps	226 Kbps	4	0	0

For the set 1, we decreased the bus speed to 35 Kbps for two networks and found a schedulable priority assignment using our gateway scheduling algorithm. Then, for the same speed, we made traditional scheduling for two networks separately. For the GW messages, we added the response times of two networks. We obtained 13 unschedulable messages at this speed. We applied the same procedure to the other message sets, in sets 1,3 and 5, the swapping part of the scheduling algorithm did not work. This is because unschedulability is due to non-gateway messages and for un-

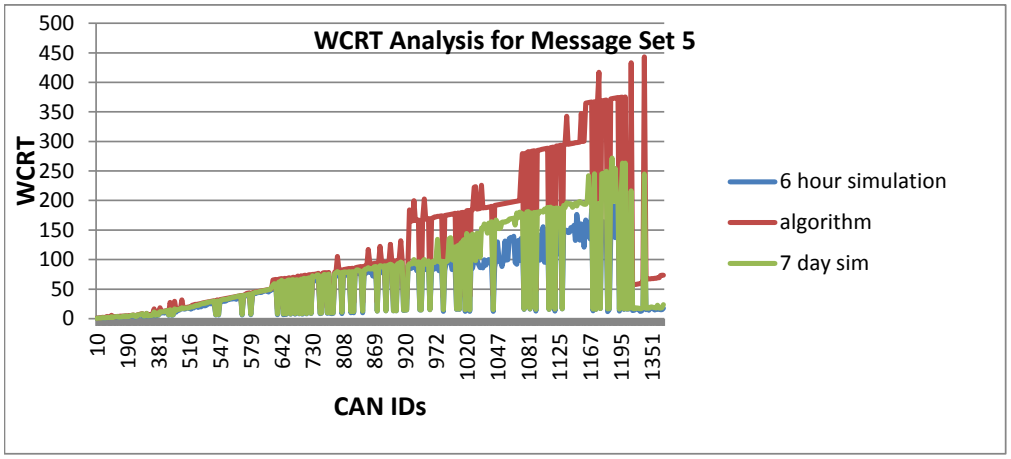

Figure 5.6: Case Study 1: WCRT analysis for message set 5

schedulable non-gateway messages nothing can be done. However, in message set 2, swapping part ran 9 times meaning that some gateway messages became schedulable by swapping the priorities. Likewise, message set 4 became schedulable at this speed after 40 swapping. With message sets 2 and 4, we tried the case in which the scheduling algorithm does not have swapping part. In that case, for message set 2, minimum bus speeds are 198 and 73 Kbps for two networks respectively. For message set 4, it s 103 Kbps for two networks. From this result, we see that swapping may help us to gain extra bandwidth.

5.7.3 Case Study 3

We conducted Case Study 3 in order to compare the performance of FIFO gateways and PQ gateways. For the 5 message sets given in Case Study 1, we made FIFO gateway analysis assuming that gateways have FIFO message queues. Then, we compared results with PQ gateway cases. The message sets and the conditions are exactly same except that in one case gateways have FIFO queues, in the other gateways has priority queues.

In Table 5.5, increase in the WCRT's of messages is shown if FIFO gateways are used instead of PQ gateways. As can be seen, increase in WCRT's of GW messages

Table5.5: Increase in the WCRT's of messages if FIFO gateways are used instead of PQ gateways

	Increase in the WCRT's of GW messages	Increase in the WCRT's of non-GW messages
Set1	5%	0%
Set2	12%	0.4%
Set3	3.3%	0.06%
Set4	17%	1.4%
Set5	10.4%	0.08%

is related to number of GW messages and bus load. There is also minor increase in the non-GW messages.

In order to show the further effects of FIFO gateways, we generated 3 message set groups. Each group has 20 message sets. Number of GW messages differs among groups. Group A has 20 GW messages, Group B has 30 and Group C has 40 GW messages. Bus loads are kept constant between 40-45% for all sets. Bus speeds are 125 Kbps for simplicity. Table 5.6 shows average WCRT values for the two cases: Gateway with priority queues and Gateway with FIFO queues. The disadvantages of FIFO queued gateways are more visible as the number of GW messages increases.

Table5.6: Average of WCRT's of GW and non-GW messages for each group for PQ and FIFO cases and Increase in the WCRT in case of FIFO queue

	PQ GW		FIFO GW		Increase in %	
	GW Messages	Non-GW Messages	GW Messages	Non-GW Messages	GW Messages	Non-GW Messages
Group A	5388	10325	6180	10488	14.7%	1.5%
Group B	10453	13375	12410	13640	18.7%	1.9%
Group C	17837	16363	21473	16986	20.3%	3.7%

CHAPTER 6

INTEGRATION IN AUTONET

6.1 Overview

Software tools developed for automotive network design are widely used in industry. Such kinds of tools provide some functions such as system configuration, schedulability test, scheduling or simulation. Existing commercial tools are expensive and not all of them use the latest research results. Moreover, they are somewhat inadequate in terms of practical restrictions or some functionalities such as signal packing. Although most of them are in capable of doing similar calculations, there is not much all-in-one tools. Therefore, we developed a tool for a project. This chapter presents the content of Project AutoNET(Automotive Network Designer). AutoNet is an automatic scheduling tool for in-vehicle networks, it is developed for TOFAS AŞ. It provides basically scheduling for CAN, for FlexRAY networks and for gateway networks. FlexRAY and gateway networks are out of the scope of this thesis . This thesis only concerns about CAN scheduling. In the CAN part of AutoNET, packing of signals, the schedulability analysis for given CAN priority assignments and the computation of CAN priority assignments are studied. Hereby, several practical constraints such as the occurrence of bus errors, the usage of FIFO queues instead of priority queues and the extension of existing CAN message sets are included in that tool. Moreover,it supports offsets and have gateway facility. All of the work done in this thesis is included in that tool. We implemented all methods in the form of a C++ software library. The implemented algorithms have also a user-friendly interface. This chapter explains the integration of developed methods and algorithms to AutoNET, introduces the graphical user interface(GUI) design and discusses the

general GUI structure and the interfaces to the underlying software library.

6.2 CAN related Algorithms

In this thesis, all of the algorithms used in AutoNET cannot be described because of limited pages. AutoNET implements the algorithms developed in this thesis, but it has also some other algorithms which were not mentioned in the previous chapters. We shortly introduce them. We implemented all methods and algorithms of AutoNET inside a class called `CANScheduler`. Here, we present related functions.

6.2.1 Classical CAN Scheduling

In the Background chapter of this thesis, we introduced classical CAN scheduling calculations. This part implements the classical CAN network analysis. It requires methods for the *WCRT* analysis of CAN messages and methods for the *assignment* of unique priorities to CAN messages in order to meet the relevant CAN performance requirements. The relevant function for WCRT analysis is `Analyze()` of the class `CANScheduler`.

The classical CAN scheduling algorithm is implemented in the CAN library, the relevant function is `Schedule()` of the class `CANScheduler`.

A feasible priority order guarantees that none of the messages violates its deadline. However, there is no additional information about further properties of the priority order. For example, the analysis does not consider that the worst-case response times of some messages might be very close to their deadline or that a single fault in a message transmission might cause deadline violation if the message has to be retransmitted. This issue is studied in [24, 21], where *robust priority orders* according to different criteria are proposed. The basic idea of this work is to consider additional *interference* (for example due to faulty transmission) that can block the transmission of messages.

The WCRT computation with additional interference and with maximum error recovery is implemented in AutoNET. The relevant functions are `AnalyzeMessage-`

`Interference()` and `AnalyzeMessageError()`.

Different versions of the robust priority assignment algorithm are implemented in the CAN library that is developed in the scope of this project. The relevant functions are `MaxErrorSchedule()` (maximizing the number of tolerated errors) `MaxInterferenceSchedule()` (maximizing the amount of tolerated interference) and `MaxSlackSchedule()` (maximizing the amount of slack) of the class `CANScheduler`.

Message set extension is another feature of AutoNET. It is frequently necessary to extend an existing message set by new messages. In this case, we want the old messages keep their original ID's and only assign priorities to new messages. The [39] paper introduces that idea. The function implemented for this method is `RobustScheduler()`.

6.2.2 CAN Networks with Practical Limitations

In FIFO Analysis chapter, we have studied systems consisting of both priority and FIFO queued controllers. The schedulability computation for CAN networks with FIFO queues is implemented in the CAN library as `AnalyzeFQAndPQ()`, `AnalyzeFIFOMessage` and `AnalyzePQMessage` functions.

In FIFO Analysis chapter, we have also discussed a scheduling algorithm for mixed systems. The relevant function for FIFO scheduling is `ScheduleFQAndPQ()` of the class `CANScheduler`.

Final algorithm we developed in FIFO Analysis chapter is FIFO extension scheduling. We have developed a message set extension algorithm for mixed systems and implemented in the CAN library as `ExtensionSchedulerForFIFO()` funtion.

6.2.3 CAN Networks with Offsets

In the CAN Offset Analysis chapter, we have seen several algorithms: The relevant function for exact offset analysis is `ExactOffsetSchedulingAnalysis()` . `OffsetWCRTAnalysis()` is the related funtion for approximate offset WCRT analysis. For maximum interference theorem analysis, we have implemented `MIFAnaly-`

`sis()` function. There are three offset assignment algorithms in that chapter: SOSA, GAOS and LNSA. The relevant functions for these algorithms are `OffsetSchedule()`, `GeneticSchedule()` and `HeuristicSchedule()`, respectively. We have also developed an algorithm for assigning offsets and priorities together. The relevant function is `OffsetandPriorityScheduleTogether()`.

6.2.4 CAN-to-CAN Gateways

In Gateway Analysis chapter, we have studied a WCRT analysis algorithm for a system consisting of two CAN networks connected via a gateway. If all the controllers including the gateway uses priority queues, then one can use `GatewayInterconnectedAnalyze()` function implemented in the CAN library. If both priority queues and FIFO queues are used, then `GatewayInterconnectedAnalyzeForFIFO()` function should be used. For priority queued networks with gateway, we have also a scheduling algorithm. The relevant function is `GatewaySchedule()`.

6.3 Graphical User Interface (GUI) Realization

The algorithms developed and described in the previous chapters are implemented in the form of a C++ software library. In order to make the algorithms conveniently available, we are developing a graphical user interface (GUI) for in-vehicle network scheduling for CAN and FlexRay. In this section, we briefly outline the organization and features of our GUI application. We are only interested in CAN part. The AutoNET GUI for CAN scheduling is realized with 5 tabs of operation: Signals and Messages, Nodes, Topology, Analysis, and Scheduling.

6.3.1 Description

The Graphical User Interface (GUI) of AutoNET is designed to decrease the time required by the engineers who design the vehicle network. Furthermore, the possible design errors can be mitigated. The proposed GUI software architecture is presented in Fig.6.1.

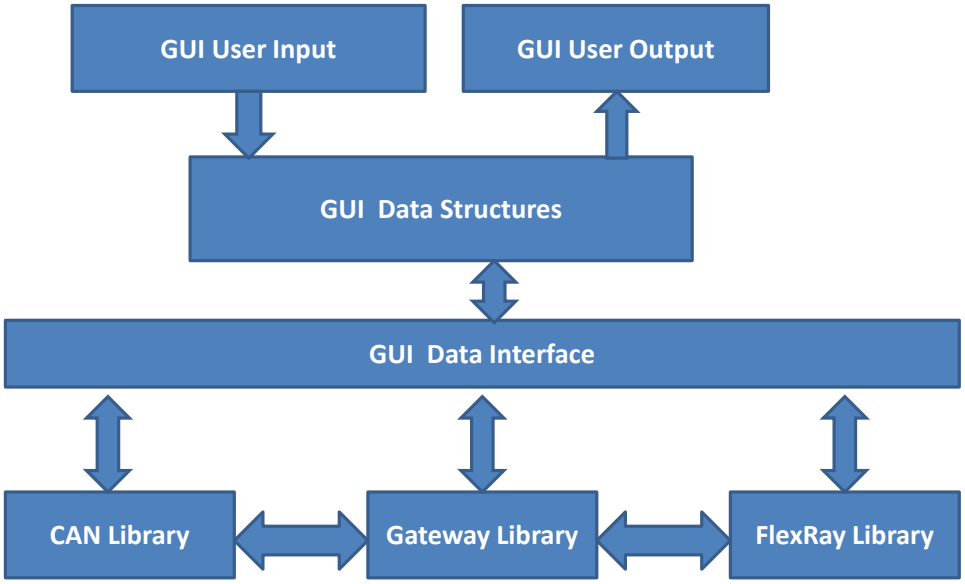

Figure 6.1: Graphical User Interface Architecture

The GUI provides input and output interfaces to the users. The inputs from the users and the outputs that are produced by the underlying CAN, FlexRay and Gateway libraries are stored in the container objects defined in the *GUI Data Structures*. Furthermore events such as click, scroll, drag and drop, menus, window controls, hierarchy and connections are defined in the GUI Data Structures. The components of the GUI Data Structure are mapped to the CAN, FlexRay and Gateway Libraries by the *GUI Data Interface*. This interface passes the inputs to the CAN, FlexRay and Gateway Libraries and the outputs (the signal message mapping, network schedules and worst case delay and jitter values) to the GUI data structure.

6.3.2 GUI User Input/Output Interfaces

The GUI user input interface is drag and drop and file inputs according to the standard formats such as FIBEX and dbc as well as any other desired format. With these formats, message sets(type of the message (periodic/sporadic) period, deadline, bit length, source node, destination node(s)), network topology, ECUs, types of networks (CAN/FlexRay), CAN bus rates, segments, gateways, and clusters are given to the tool. An example vehicle network topology is shown in Fig. 6.2.

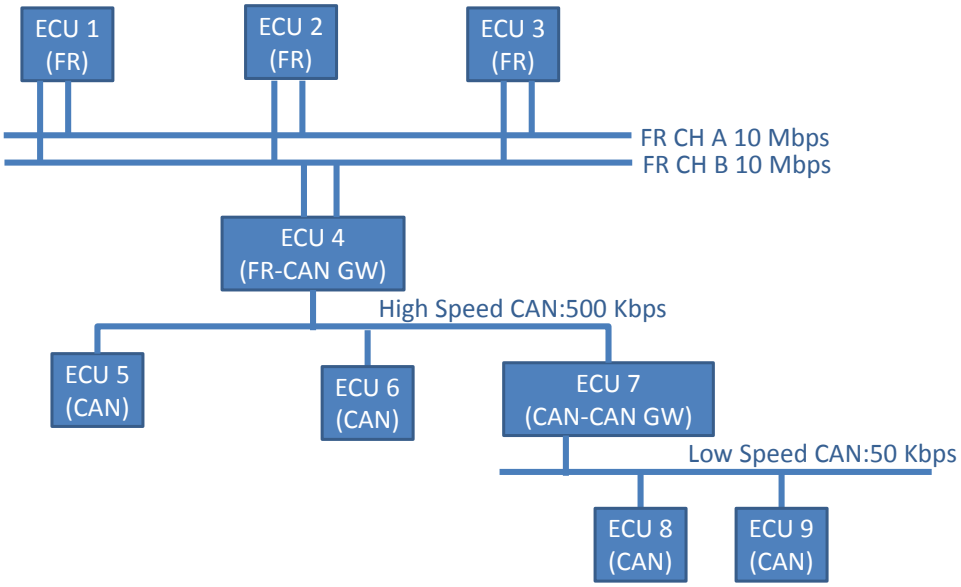

Figure 6.2: Example Vehicle Network Topology

The GUI user output is provided as tables, plots and standard configuration file formats (FIBEX, CAN dbc). The output data computed by the CAN library of AutoNET is:

- Worst case response times for the messages, deadline violations
- CAN schedules (CAN ID assignments, offset assignments),

CHAPTER 7

CONCLUSION

The recent advances in automobile technology lead to an increase in the number of messages in in-vehicle networks. As a result, several additions to the classical Controller Area Network (CAN) such as the usage of transmission offsets and the idea of message set extension have been proposed. In addition, several practical limitations of CAN such as the usage of FIFO queues (FQS) instead of priority queues (PQs) or the division of CAN networks into different segments that are connected by gateways need to be considered in practical applications. The main focus of this thesis is the worst-case response time (WCRT) analysis and the message priority assignment for CAN networks with the stated extensions and restrictions.

First, the usage of transmission offsets on CAN is considered. Three different algorithms for the WCRT computation are implemented in a C++ library and compared. It is concluded that the Maximum Interference Method (MIF) is most suitable. In addition, different methods for the offset assignment are implemented and compared by case studies. Finally, as an entirely novel contribution, an algorithm for the combined offset and priority assignment for CAN messages is proposed. The benefits of this algorithm are shown in computational experiments.

Second, CAN networks with PQs and FQs are considered. Here, a modified WCRT analysis algorithm that reduces the complexity of the existing analysis is developed. Moreover, an algorithm for extending an existing CAN application by new messages is proposed. A paper on this research has been submitted to the International Journal of Vehicle Design.

Third, CAN gateway networks are considered. The existing WCRT algorithm for such networks is implemented and its correctness is evaluated by simulation. In addition, for the first time, a priority assignment algorithm for gateway networks is developed and its efficiency is shown in computational experiments. Moreover, as a further novel contribution, the WCRT analysis on CAN gateway networks is extended to the case of CAN networks with FQs and PQs.

All the algorithms that are implemented in the scope of this thesis are made available for the use in the software tool AutoNET for in-vehicle networks.

Topics for future research can be concluded form Table 1.1. In particular, the work on CAN networks with FQs and CAN messages with offsets can be used to combine both of them. That is, FQ-Offset WCRT analysis and priority assignment is an interesting topic for future research. Furthermore, a proof of optimality of our message set extension algorithm for CAN networks with FQs and PQs is a further task for future work. Finally, it is envisaged to extend the priority assignment algorithm for CAN gateway networks that is developed in this thesis to the case of CAN networks with FQs and PQs.

REFERENCES

[1] *CAN BUS Bridge(Intelligent Repeater)*, (accessed December 11, 2014).

[2] *CAN Series Products*, (accessed December 11, 2014).

[3] *CANbridge for CAN and CANOpen Systems*, (accessed December 11, 2014).

[4] *CAN/CAN Gateway CG0ARM7*, (accessed December 11, 2014).

[5] *PCAN Router-Peak Systems*, (accessed December 11, 2014).

[6] *What is ahead for CAN*, (accessed November 3, 2014).

[7] *NETCAR-Analyzer : the RTaW-Sim plugin for worst-case timing analysis on Controller Area Network*, (accessed October 10, 2014).

[8] *NETCARBENCH A benchmark generator for automotive communication systems*, (accessed October 10, 2014).

[9] *RTaW-Sim : Controller Area Network simulation and configuration*, (accessed October 10, 2014).

[10] B. Alkan, E. G. Schmidt, and K. W. Schmidt. Karayollarında akıllı ulaşım sistemleri için araç içi ağları. In *Karayolu Akıllı Ulaşım Sistemleri Kongre ve Sergisi*, 2014.

[11] B. Alkan, E. G. Schmidt, K. W. Schmidt, D. Çulum Karani, and U. Karakaya. Autonet - an automatic scheduling tool for in-vehicle networks. In *OTEKON*, 2014.

[12] B. Alkan, E. G. Schmidt, K. W. Schmidt, D. Çulum Karani, and U. Karakaya. Controller area network (can) with priority queues and fifo queues: Improved schedulability analysis and message set extension. *International Journal of Vehicle Design (submitted)*, 2014.

[13] N. Audsley. Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. *Technical Report YCS 164, Dept. Computer Science, University of York, UK*, 1991.

[14] N. Audsley. On priority assignment in fixed priority scheduling. *Information Processing Letters*, 79(1):39–44, 2001.

[15] N. Audsley. On priority assignment in fixed priority scheduling. *Information Processing Letters*, 79(1):39–44, 2001.

[16] V. Celik. Development of strategies for reducing the worst-case message response times on controller area network. Master's thesis, Middle East Technical University, 2012.

[17] Y. Chen. Real time scheduling and analysis for can messages with offsets. Master's thesis, Nagoya University, 2012.

[18] R. B. D. Khan and N. Navet. Integrating hardware limitations in can schedulability analysis. In *in Factory Communication Systems (WFCS)*, pages 207–210, 2010.

[19] R. Davis, A. Burns, R. Bril, and J. Lukkien. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Syst.*, 35(3):239–272, April 2007.

[20] R. I. Davis and A. Burns. Robust priority assignment for fixed priority real-time systems. *IEEE International Real-Time Systems Symposium*, pages 3–14, 2007.

[21] R. I. Davis and A. Burns. Robust priority assignment for fixed priority real-time systems. In *IEEE International Real-Time Systems Symposium*, pages 3–14, 2007.

[22] R. I. Davis and A. Burns. Robust priority assignment for messages on controller area network (CAN). *Real-Time Syst.*, 41(2):152–180, Feb. 2009.

[23] R. I. Davis and A. Burns. Robust priority assignment for messages on controller area network (can). *Real-Time Syst*, 41(2):152–180, 2009.

[24] R. I. Davis and A. Burns. Robust priority assignment for messages on controller area network (CAN). *Real-Time Syst.*, 41(2):152–180, Feb. 2009.

[25] R. I. Davis, S. Kollmann, V. Pollex, and F. Slomka. Schedulability analysis for controller area network (CAN) with FIFO queues priority queues and gateways. *Real-Time Syst.*, 49(1):73–116, Jan. 2013.

[26] L. Du and G. Xu. Worst case response time analysis for CAN messages with offsets. In *Vehicular Electronics and Safety, IEEE International Conference on*, pages 41 –45, 2009.

[27] P. E. Ekiz H., Kutlu A. Design and implementation of a can2can bridge. In *Proceedings of IEEE ISPAN96 Conference*, 1996.

[28] P. E. Ekiz H., Kutlu A. Implementation of can / can bridges in distributed environments and performance analysis of bridged can systems using sae benchmark. 1997.

[29] I. B. F. Polzlbauer and E. Brenner. Optimized frame packing for embedded systems. *Embedded Systems Letters, IEEE*, 4:65–68, 2012.

[30] L. George, N. Rivierre, and M. Spuri. Pre-emptive and non-pre-emptive real-time uni-processor scheduling. *Technical Report 2966, Institut National de Recherche et Informatique et en Automatique (INRIA), France*, 1996.

[31] M. Grenier, L. Havet, and N. Navet. Scheduling messages with offsets on Controller Area Network - a major performance boost. In *The Automotive Embedded Systems Handbook*, Industrial Information Technology Series. Taylor & Francis / CRC Press, 2008.

[32] U. Keskin. In-vehicle communication networks: A literature survey. Master's thesis, Technische Universiteit Eindhoven (TU/e) Den Dolech 2, 5600 AZ Eindhoven, The Netherlands, July 28, 2009.

[33] 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*, pages 1–10, 2012.

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

[35] M. D. Natale. Evaluating message transmission times in controller area networks without buffer preemption. In *8th Brazilian Workshop on Real-time Systems*, 2006.

[36] M. D. Natale. *Understanding and using the Controller Area network*, (accessed October 10, 2014).

[37] T. Nolte. Share-driven scheduling of embedded networks. Master's thesis, Malardalen University Press Dissertations, No. 26. Department of Computer Science and Electronics, Malardalen University, Sweden, May 2006.

[38] N. N. R.I. Davis. Controller area network (can) schedulability analysis for messages with arbitrary deadlines in fifo and work-conserving queue. In *9th workshop on factory communication systems*, pages 33–42, 2012.

[39] K. Schmidt. Robust priority assignments for extending existing controller area network applications. *Industrial Informatics, IEEE Transactions on*, 10(1):578–585, 2014.

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

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

[42] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming*, 50(2–3):117–134, 1994.

[43] A. Valenzano and G. Cena. Controller area networks for embedded systems. In R. Zurawsk, editor, *Networked Embedded Systems*, pages 15–1–15–38. CRC Press, 2009.

[44] H. T. Y. Chen, R. Kurachi and G. Zeng. Schedulability comparison for can message with offset: Priority queue versus fifo queue. *RTNS*, pages 181–192, 2011.

[45] Y. C.-R. K. H. T. Y. Xie, G. Zeng and R. Li. Schedulability analysis for messages in gateway-interconnected controller area network. *International Conference on Connected Vehicles and Expo*, 2012.

[46] Y. C.-R. K. H. T. Y. Xie, G. Zeng 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.

[47] 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*, pages 43–52, 2012.