TIME-TRIGGERED CONTROLLER AREA NETWORK (TTCAN) COMMUNICATION SCHEDULING: A SYSTEMATIC APPROACH

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

BY

UĞUR KESKİN

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

AUGUST 2008

Approval of the thesis:

TIME-TRIGGERED CONTROLLER AREA NETWORK (TTCAN) COMMUNICATION SCHEDULING: A SYSTEMATIC APPROACH

submitted by **UĞUR KESKİN** 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. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen
Head of Department, **Electrical and Electronics Engineering**

Assist. Prof. Dr. Ş. Ece Schmidt
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Semih Bilgen
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Ş. Ece Schmidt
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering Dept., METU

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

Uğur Kazancıoğlu (M.S.)
SST, ASELSAN

Date: 22.08.2008

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 : Uğur Keskin

Signature :

ABSTRACT

TIME-TRIGGERED CONTROLLER AREA NETWORK (TTCAN) COMMUNICATION SCHEDULING: A SYSTEMATIC APPROACH

Keskin, Uğur
M.S., Department of Electrical and Electronics Engineering
Supervisor: Assist. Prof. Dr. Ş. Ece Schmidt

August 2008, 103 pages

Time-Triggered Controller Area Network (TTCAN) is a hybrid communication paradigm with combining both time-triggered and event-triggered traffic scheduling. Different from the standard Controller Area Network (CAN), communication in TTCAN is performed according to a pre-computed, fixed (during system run) schedule that is called as TTCAN System Matrix. Thus, communication performance of TTCAN network is directly related to structure of the system matrix, which makes the design of system matrix a crucial process. The study in this thesis consists of the extended work on the development of a systematic approach for system matrix construction. Methods for periodic message scheduling and an approach for aperiodic message scheduling are proposed with the aim of constructing a feasible system matrix, combining three important aspects: message properties, protocol constraints and system performance requirements in terms of designated performance metrics. Also, system matrix design, analyses and performance evaluation are performed on example message sets with the help of two developed software tools.

Key words: In-vehicle Real-time Embedded Systems, Vehicular Communication Networks, Time-Triggered Controller Area Network (TTCAN), Communication Scheduling, Real-time Communication Performance Analysis

ÖZ

ZAMANLA TETİKLENEN DENETLEYİCİ ALAN AĞINDA (TTCAN) HABERLEŞME ÇİZELGELENMESİ: SİSTEMATİK BİR YAKLAŞIM

Keskin, Uğur
Yüksek Lisans, Elektrik-Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Yrd. Doç. Dr. Ş. Ece Schmidt

Ağustos 2008, 103 sayfa

Zamanla tetiklenen denetleyici alan ağı (TTCAN), zamanla tetiklenen trafikle birlikte, olayla tetiklenen haberleşme trafiğini de destekler. Standart denetleyici alan ağından (CAN) farklı olarak, TTCAN ağında haberleşme, sistem koşumundan önce hazırlanmış ve değişmez bir sistem matrisine dayanarak yürütülmektedir. Bir TTCAN ağının gerçek zamanlı haberleşme performansı sistem matrisi yapısına birebir bağlıdır. Bu nedenle sistem matrisinin oluşturulması çok önemli bir süreçtir. Bu tez, sistem matrisi oluşturulmasında sistematik bir yaklaşım üzerine devam çalışmalarını içermektedir. Sistem matrisi oluşturulmasında düşünülmüş üç önemli başlık olan; mesaj özellikleri, belirlenmiş performans metrikleri cinsinden sistem performans gerekleri ve protokol sınırları birleştirilerek uygun bir sistem matrisinin oluşturulması kapsamında periyodik mesajların çizelgelenmesi için metotlar ve periyodik olmayan mesajların çizelgelenmesi için bir yaklaşım önerilmiştir. Ayrıca, geliştirilen iki yazılım aracıyla, örnek mesaj setleri üzerinde sistem matrisinin kurulması, analizler ve haberleşme performansı değerlendirilmesi yapılmıştır.

Anahtar Kelimeler: Araç İçi Gerçek Zamanlı Gömülü Sistemler, Araçsal İletişim Ağları, Zamanla Tetiklenen Denetleyici Alan Ağı, İletişim Çizelgelenmesi, Gerçek Zamanlı Haberleşme Performans Analizi

To Mom, Dad, Sisters, Nieces and Nephews

ACKNOWLEDGEMENTS

I would like to express my gratitude and deep appreciation to my supervisor Assist. Prof. Dr. Ş. Ece Schmidt for his guidance, positive suggestions and also for the great research environment she had provided.

I would like to also express my thanks for their great friendship and assistance to Hasan Akyol, Deniz Özcan, Oytun Akman and Taha Ahi.

I would like to thank my co-workers in MODSİMMER and Assoc. Prof. Dr. Veysi İşler and Utkan Eryılmaz for such a friendly research environment they had provided. I have learned much from their suggestions and experiences.

Finally, I would like to thank my Mom, Dad and Sisters for their love, support and patience over the years. This thesis is dedicated to them.

TABLE OF CONTENTS

ABSTRACT iv
ÖZ vi
ACKNOWLEDGEMENTS ix
TABLE OF CONTENTS x
LIST OF TABLES xii
LIST OF FIGURES xiv
LIST OF ABBREVIATIONS xv
CHAPTER
1. INTRODUCTION 1
1.1 Scope of the Thesis 3
1.2 Outline of the Thesis 4
2. CAN AND TTCAN COMMUNICATION PROTOCOLS 5
2.1 In-Vehicle Communication 5
2.2 Controller Area Network (CAN) 11
2.2.1 CAN Message Format 12
2.2.2 CAN Arbitration 14
2.2.3 CAN Priority Scheduling 16
2.2.3.1 Fixed (Static) Priority Scheduling 17
2.2.3.2 Dynamic Priority Scheduling 18
2.2.4 Drawbacks of CAN 19
2.3 Time-Triggered Controller Area Network (TTCAN) 20
2.3.1 TTCAN Communication Scheduling: System Matrix 21
2.3.2 Related Work on TTCAN Communication Scheduling 24
3. TTCAN COMMUNICATION SCHEDULING COMPONENTS 27
3.1 Background Information 27
3.1.1 Assumptions 28

3.1.2 Message Set Properties..29

3.1.3 Performance Metrics 32

3.2 Protocol and Hardware Constraints..34

3.3 Additional Performance Metrics 35

4. TTCAN MESSAGE SCHEDULING: SYSTEM MATRIX DESIGN 39

4.1 Problem Formulation and Contribution of the Thesis 40

4.2 Background Information on Periodic Message Scheduling............42

4.3 TTCAN Scheduler Tool...48

4.4 SM Design for Periodic Message Set 48

4.4.1 SM Design for Ideal Message Set..49

4.4.2 SM Design for Non-Ideal Message Set.................................61

4.5 SM Design for Sporadic Messages..76

5. TEST RESULTS BY TTCAN SIMULATION 83

5.1 TTCAN Simulation Tool 83

5.2 Application Examples...86

6. CONCLUSION 93

6.1 Summary of the Thesis 93

6.2 Future Work 97

REFERENCES...98

LIST OF TABLES

TABLES

Table 2.1 Comparison of event and time-triggered communication paradigms .. 10

Table 2.2 An example relating the trigger and window properties of a TTCAN node based on the SM in Figure 2.5 23

Table 3.1 PSA benchmark periodic message set 31

Table 4.1 SAE benchmark periodic message set 46

Table 4.2 A self constructed non-ideal message set example 68

Table 4.3 The SM to provide minimum number of triggers with the related performance results for the non-ideal message given in Table 4.2............... 69

Table 4.4 The SM after doubling *B* to provide less bandwidth loss and related performance results 70

Table 4.5 The SM after further period reduction to provide less jitter and related performance results 72

Table 4.6 SM for SAE periodic message set after period reduction and related performance results 73

Table 4.7 SM for SAE periodic message set after doubling *B* and related performance results 74

Table 4.8 SM for SAE periodic message set for less trigger count and jitter and related performance results.. 75

Table 4.9 SAE benchmark sporadic message set 76

Table 5.1 Performance results of the SM given in Figure 5.1 under two different arrival traffics of SAE sporadic messages... 86

Table 5.2 Performance results of the SM given in Figure 5.1 with using two approaches for sporadic message scheduling for the SAE sporadic message set 87

Table 5.3 Performance results for the system matrices with two different arrangements of arbitration columns as given by Figure 5.2......................... 90

Table 5.4 Performance results for the system matrices given by Figure 5.3 92

LIST OF FIGURES

FIGURES

Figure 2.1 Comparison of some in-vehicle network protocols with respect to data rate and communication cost [8] ... 7

Figure 2.2 Fieldbus network architecture ... 8

Figure 2.3 CAN 2.0A message format ... 13

Figure 2.4 An example relating CAN arbitration between three nodes ... 15

Figure 2.5 TTCAN system matrix ... 22

Figure 4.1 An example relating SM design for an ideal message set with equal message lengths ... 44

Figure 4.2 Implementation of the method to obtain optimum SM design for an ideal message set with unequal messages lengths ... 52

Figure 4.3 Optimum SM construction to provide minimum in-window loss for an example periodic message set ... 57

Figure 4.4 The SM designed by using the first approach for PSA benchmark message set ... 58

Figure 4.5 The SM designed by using the second approach for PSA benchmark message set ... 59

Figure 4.6 The SM designed by using the second approach with increased duration allocated to periodic message scheduling ... 60

Figure 4.7 An example for arbitration window arrangement ... 79

Figure 5.1 An example of a SM input file for TTCAN simulation tool ... 84

Figure 5.2 The SM with two different arbitration column arrangements ... 89

Figure 5.3 The SM with four different arbitration arrangements ... 91

LIST OF ABBREVIATIONS

ABS	Antilock Braking System
ACC	Adaptive Cruise Control
ACK	Acknowledgment
ASIC	Application Specific Integrated Circuit
ASR	Anti-Slip Regulation
BC	Basic Cycle
CAN	Controller Area Network
CO	Cycle Offset
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
D2B	Digital Data Bus
DLC	Data Length Code
DLL	Data Link Layer
DM	Deadline Monotonic
EBD	Electronic Brakeforce Distribution
ECU	Electronic Control Unit
EDF	Earliest Deadline First
EEPROM	Electrically Erasable Programmable Read-Only Memory
EOF	End of frame
ESP	Electronic Stability Program
FTT-CAN	Flexible Time-Triggered Controller Area Network
I/O	Input/Output
LIN	Local Interconnect Network
MOST	Media-Oriented System Transport
MTS	Mixed Traffic Scheduler

NRC	Non-Return-to-Zero
NTU	Network Time Unit
PSA	Peugeot-Citroen Automobiles Company
RAM	Random Access Memory
RF	Repeat Factor
RM	Rate Monotonic
ROM	Read-Only Memory
RTR	Remote Transmission Request
SAE	Society of Automotive Engineers
SM	System Matrix
SOF	Start of Frame
TDMA	Time Division Multiple Access
TTCAN	Time-Triggered Controller Area Network
TTP	Time-Triggered Protocol
VAN	Vehicle Area Network

CHAPTER 1

INTRODUCTION

Real-time systems are special purpose computing systems that are designed to perform assigned functions with timing constraints correctly [1-3]. Real-time embedded systems range from small, portable devices to large and complex structures and they are responsible for time-critical applications. In-vehicle systems are such structures that widely comprise real-time embedded systems. Electronic systems have been increasingly replacing old in-vehicle mechanical and hydraulic counter-parts for last more than 30 years with the aim of comfort and safety [4]. For instance, in today's modern automobiles electronic systems such as, antilock braking system (ABS) and electronic brakeforce distribution (EBD), electronic power steering (EPS), anti-slip regulation (ASR), electronic stability program (ESP) and adaptive cruise control (ACC) assist the driver in control with providing more comfort and safety. Similarly, devices that belong to body domain of a car (dashboard, lights, wipers, doors and windows) as well as communication and entertainment equipment (radio, navigation systems, mobile phones) are controlled by electronic systems [4].

Increasing user requirements and technology development result in more complex in-vehicle electronic systems; obtaining higher number of electronic control units (ECU) and communication signals with more complex interrelations between them. This result reveals the need for more robust, dependable and efficient high speed in-vehicle communication. Real-time performance is very important for electronic systems related to braking, steering, engine control and safety domains, that is; they can be seen as real-time embedded systems that perform functions with strict timing constraints. Such systems also contain hard real-time messages that require strict timing requirements. The exchange of these messages in the network is conducted by the in-vehicle communication protocols that can be classified as event-triggered, time-triggered and hybrid networks. These networks are expected to schedule real-time messages to provide timeliness in communication for healthy run of the system. In event-triggered networks, messages are transmitted upon the occurrences of significant events and access to the bus is granted based message priorities. In time-triggered networks, communication between nodes is performed by the progress of time. Hybrid networks contain both event and time-triggered traffics with satisfying temporal isolation between them.

Controller Area Network (CAN) that is an event-triggered protocol is the most widely used in-vehicle communication network. As a time-triggered version of CAN, Time-Triggered Controller Area Network (TTCAN) has important advantages of dependability and predictability compared to CAN. TTCAN network contains both event and time triggered traffic with satisfying temporal isolation between them by combining the advantages of both communication classes. Also, In TTCAN networks, time windows are reserved for hard real-time messages based on time division multiple access (TDMA) bandwidth allocation scheme. Yet, this necessitates a schedule that manages the exchange of both event and time triggered traffic. Because of that, real-time communication performance of a TTCAN network highly depends on this schedule. The thesis consists of extension of the work in [5], which proposes a systematic approach on TTCAN communication scheduling. The main theme of the thesis is on the construction of the TTCAN communication schedule considering protocol constraints, message properties and communication performance requirements of the system.

1.1 Scope of the Thesis

The main theme of the thesis is on the construction of the TTCAN communication schedule considering protocol constraints, message properties and communication performance requirements of the system.

This thesis can be investigated under three main parts; background, TTCAN communication scheduling and finally test results by TTCAN simulation. The first part is a literature survey providing background information about both CAN and TTCAN networks that is necessary for better understanding of the following chapters. Background part relates not only technical specifications of the protocols but also principles and properties that make them so popular for in-vehicle data transmission and the motivation behind the preference of TTCAN as a communication protocol. In addition, different approaches in previous studies on TTCAN communication scheduling are discussed in the first part.

The second part focuses on the design of TTCAN communication schedule, where the referred and proposed methods and algorithms are explained to obtain a feasible TTCAN schedule for both event and time-triggered traffic by combining three important concepts that are message properties, protocol constraints and system requirements in terms of performance metrics. Yet, a feasible schedule with respect to what? So at first, the assumptions, message set properties and performance metrics are explained, which form the basis of the SM design. The scheduling approach for real-time messages is explained with providing methods and analyses.

And finally the third part includes simulation results based on different configurations of schedules. The designed TTCAN schedules are tested including event-triggered traffic.

1.2 Outline of the Thesis

This thesis is divided into five main chapters as background, TTCAN communication scheduling components, TTCAN message scheduling: SM design, test results by TTCAN simulation and finally conclusion.

In Chapter 2, background information on CAN and TTCAN communication networks is given. They are compared in working principles and communication performance, and the motivation behind preferring TTCAN is provided. Also different approaches in previous studies on TTCAN scheduling are discussed.

Chapter 3 relates the designated performance metrics and message properties including the relations between them. Moreover, hardware and protocol constraints are explained in the chapter.

Chapter 4 discusses the TTCAN message scheduling as the construction of the TTCAN schedule and extends the systematic approach with providing methods and analyses.

In Chapter 5, the simulation tool is explained and test results are given based on the simulation of designed schedules with focusing on event-triggered traffic.

Finally, the thesis ends with summary and foreseen future work plan in Chapter 6.

CHAPTER 2

CAN AND TTCAN COMMUNICATION PROTOCOLS

Chapter 2 will provide background material about in-vehicle communication and the protocols, Controller Area Network (CAN) and Time-Triggered Controller Area Network (TTCAN), which is necessary for better understanding of the following chapters. The first section of the chapter will provide an overview on communication paradigms used in vehicles. The second section will give general and technical specifications of CAN bus in addition to arbitration (scheduling) policies applied on CAN. Similarly, in the second section TTCAN protocol will be explained and also previous studies on TTCAN communication scheduling will be discussed.

2.1 In-Vehicle Communication

In-vehicle embedded systems could be divided into several functional domains based on corresponding properties such as, architectures, services and constraints; *powertrain* (i.e. engine control), *chassis* (i.e. control of steering and braking), *body* and *telematics* in addition to an emerging domain *safety* (ACC, impact and rollover sensors and airbags) [4][14]. Among these, powertrain and chassis domains are more concerned with real-time control and safety; while body domain mainly implements comfort functions. Telematics domain integrates multimedia, monitoring, communication, navigation and entertainment facilities. Because of such diverse properties and functions, in-vehicle domains possess different communication requirements. In 1994, Society of Automotive Engineers (SAE) defined a classification for in-vehicle networks based on their function and bandwidth (data rate) requirements [4][6][7]. In this classification, *Class A* denote low speed/low cost networks with data rate less than 10 kb/s and they are mostly dedicated to body domain. Local Interconnect Network (LIN) [8] and Time-Triggered Light Weight Protocol (TTP/A) are examples of such networks [4]. *Class B* networks, operating between 10 and 125 kb/s data rates, are used for general information exchange (i.e. vehicle speed, instrument cluster). J1850 [9] and low speed Controller Area Network (CAN-B) are the main examples of this class. Different from above, *Class C* (i.e. high speed CAN (CAN-C) [10]) and *Class D* networks require high speed communication. Data rate for *Class C* networks range from 125 kb/s to 1Mb/s and used for a wide range of applications especially in powertrain and chassis domains. On the other hand, data rate in *Class D* networks is up to or higher than 1 Mb/s, where they are mainly used for multimedia data and x-by-wire applications [4]. Media-Oriented System Transport (MOST) [11], Digital Data Bus (D2B) [12] and Bluetooth as wireless communication [13] are main examples of Class D networks for multimedia data transmission. In addition, the term x-by-wire [14] represents in-car critical and safety related applications that have strict timing constraints and it is used in such forms as brake-by-wire and steer-by-wire etc. Being dependable, robust with high speed, Time Triggered Protocol (TTP/C) [15], FlexRay [16] and Byteflight [13][17] protocols have been proposed as appropriate solutions for x-by-wire applications.

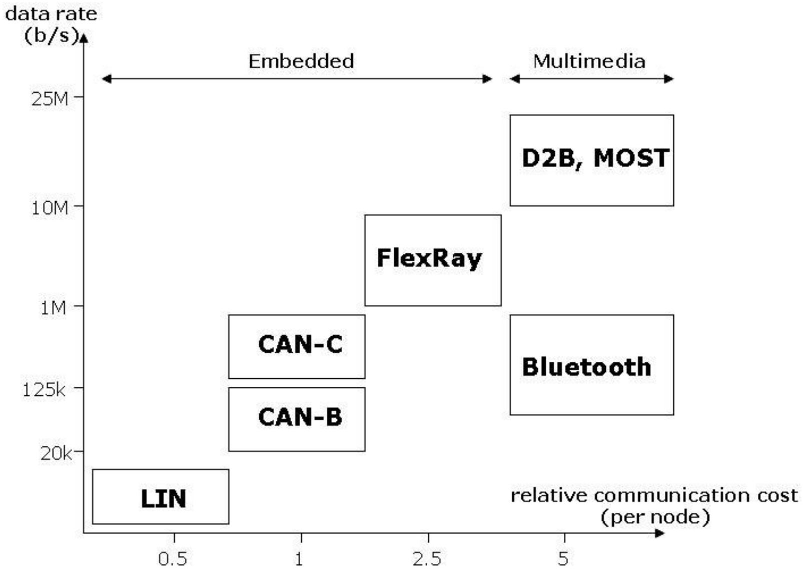


Figure 2.1 Comparison of some in-vehicle network protocols with respect to data rate and communication cost [8]

In Figure 2.1 [8], some of above stated protocols are placed in the graphic based on their allowable data rates with respect to relative communication cost per ECU (node). In general, wiring, microcontrollers and other hardware implementation as well as data overhead and resource consumption determine the cost value [8][11]. Yet, comparison of different protocols is also possible considering performance properties apart from data rate, such as flexibility, efficiency, robustness and dependability. The following section mainly focuses on this discussion.

In today's modern vehicles nearly 2500 signals are exchanged by up to 70 electronic control units (ECU); that is, embedded systems in cars are complex, networked and distributed structures [18]. Moreover, Increasing electronic systems and application requirements in vehicles reveal the need not only for smaller and higher performance microcontrollers as ECUs but also for high speed, low cost, robust and efficient communication networks with minimum wiring. Several network topologies (i.e. mesh, star, bus and topologies with gateways etc.) would be proposed to provide communication between networked nodes. At this point, because of being simple and versatile (easy system extension and evolution) and having low cost (installation cost saving with less wiring) serial communication with bus networks comes up as an appropriate solution [19]. Figure 2.2 illustrates a Fieldbus network architecture example comprising a bus and nodes (ECUs) that each node consists of a central processing unit (CPU), memory (RAM, ROM and EEPROM etc.), I/O interface and communication interface. Also, nodes are possible to obtain application specific integrated circuits (ASIC) for acceleration purpose [3].

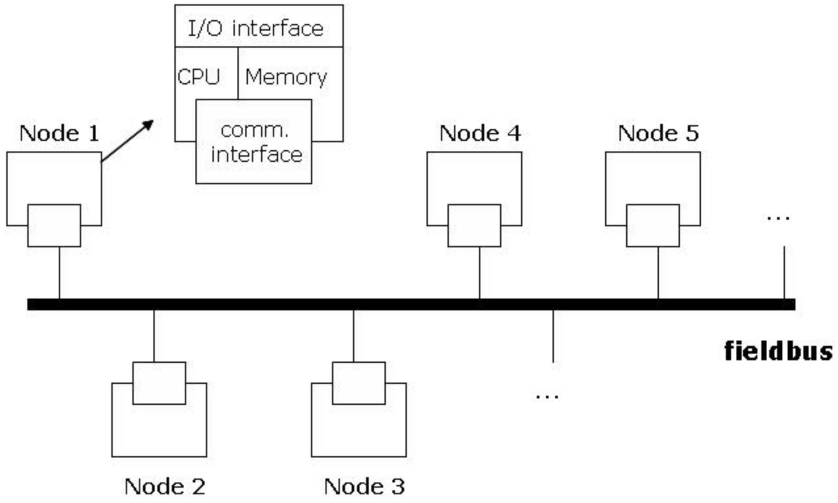


Figure 2.2 Fieldbus network architecture

Bus network protocols can be evaluated under different communication classifications, such as static versus dynamic, synchronous versus asynchronous, deterministic versus nondeterministic and finally time-triggered versus event-triggered [20]. Although these paradigms are based on similar concepts, in literature time versus event triggered classification is the most commonly contrasted one [18][20-22].

In event-triggered communication messages are transmitted based on significant events and asynchronous (event-triggered) message transmissions are performed as quickly as possible [4]. Most event-triggered protocols are based on CSMA/CA (carrier sense multiple access/collision avoidance) media access method [23]. Transmission of messages is performed by bus arbitration based on message priorities to prevent collisions. Because of this property, such bus networks are also called as priority busses [4]. Flexibility, convenience to easy system change and evolution and ability of quick response to asynchronous events are important advantages of event-triggered approach. Especially, quick response and message transmission upon occurrences of events make the paradigm have higher flexibility (flexible to varying network traffics) and bandwidth efficiency. Vehicle Area Network (VAN) [24], J1850 and CAN are the main examples of this paradigm. VAN and J1850 protocols were generally used in body domain but recently they have been replaced by CAN in vehicular communication [4].

In time-triggered approach, communication between nodes is performed by the progress of time. In other words, message transmission is driven at predefined time instants based on time division multiple access (TDMA) bandwidth allocation scheme. Since time intervals for message transmissions (access of nodes to bus) are predefined and deterministic, missing messages in the networked system or an error/fault in a node can easily be detected and removed that makes the approach predictable (bounded response times) and dependable. Yet, this property also makes system change such a hard work that adding new nodes and messages to the system results in the need for the change of the predefined communication schedule. As a result, time-triggered protocol can be proposed as an appropriate solution especially for hard real-time systems (i.e. for x-by-wire applications) that require high real-time performance with dependability, robustness and fault tolerance. For instance, TTP/C protocol implements the TDMA scheme in which each node has right to access to bus in sequential, predefined and static time instants during consecutive TDMA rounds. Consecutive TDMA rounds form the cluster cycle that repeats itself in a loop during the system run.

Table 2.1 summarizes the positive and negative aspects of event and time-triggered communication. The term "resource efficiency" in the table implies the utilization of bandwidth with message transmissions between nodes. Also, the term "system change" refers to software or hardware evolution and extension of a networked embedded system such as, adding new nodes or messages to system.

Table 2.1 Comparison of event and time-triggered communication paradigms

Event-Triggered Communication (CAN, VAN and J1850)	
Advantages	**Disadvantages**
- resource efficient - flexible - convenient to easy system change	- low predictability - low dependability - low fault tolerance - unverified response delays
Time-Triggered Communication (TTP/C)	
Advantages	**Disadvantages**
- fully predictable - bounded response delay - no need for arbitration policy (simplicity) - high dependability and fault tolerance	- low resource efficiency - low flexibility - difficult system change

Similar to time-triggered paradigm, communication protocols such as Time-Triggered Controller Area Network (TTCAN) [25][26], Flexible Time-Triggered Controller Area Network (FTT-CAN) [20], FlexRay and Byteflight are basically implemented on TDMA bandwidth allocation scheme. Yet, such protocols also shelter event-triggered (asynchronous) messages with satisfying temporal isolation between event and time-triggered traffics. These protocols can be called as hybrid [22] as combining both event and time-triggered approaches with the aim of utilizing advantages of both. In other words, in addition to fault tolerant and dependability properties they provide higher flexibility and resource efficiency compared to time-triggered paradigm. On the other hand, the low speed protocols LIN and TTP/A conducts network communication based on master-slave procedure in which a master node in the network, that possess a global and exact time base, coordinates the message transmission between nodes. The master node sends a "command frame" before each transmission cycle to inform nodes about which message is to be transmitted. Among all stated protocols, CAN and TTCAN as a time triggered version of CAN are the most commonly used in-vehicle network protocols. In the following sections, these protocols will be discussed in the scope of communication scheduling.

2.2 Controller Area Network (CAN)

CAN is a serial, broadcast bus that is developed by Robert Bosch GmbH in mid-1980s, then it became a ISO standard in 1994, and currently it is de facto standard for in-vehicle data transmission [4][10]. CAN is the most widely used automotive communication network with the advantages of providing flexible and robust communication with bounded delay and having low cost (simplicity). At first CAN was developed specifically for in-vehicle communication to provide data transmission between nodes forming the networked system. Yet, now it is also used for real-time distributed systems in several industrial applications (i.e.

mobile robots, factory automation) and nearly 400 million CAN nodes are sold each year. CAN is implemented on twisted pair of copper wires operating at a bus data rate (bandwidth) ranging from 20 kb/s to 1 Mb/s. Message transmission on CAN depends on nodes' monitoring the bus that is, transmitted signal propagates to the most remote node in the network and then returns back. Propagation delay grows with increasing bus length; which limits data rate such as, for 1 Mb/s bus rate maximum bus length is 40-m whereas for 250 kb/s it becomes 250-m [4]. Thus, CAN is considered to be appropriate for small or medium scale networks. CAN-B networks (data rate up to 125 kb/s) are generally used in *body* domain in vehicles; on the other hand CAN-C networks (data rate up to 1 Mb/s) are mostly used for real-time control in *powertrain* and *chassis* domains. Throughout the thesis high speed CAN applications will be considered.

2.2.1 CAN Message Format

In a CAN network there exist four different frame types: data frame, remote frame, error frame and overload frame [21]. A standard CAN 2.0A data frame consists of seven fields: start of frame (SOF) bit, 18 bits header, 0-8 byte data, 15 bits cyclic redundancy check (CRC) field, 3 bits acknowledgement slot (ACK), 7 bits end of frame field (EOF) and last 3 bits intermission frame space. Moreover, header of a frame can be divided in to 3 minor fields that are 11 bits identifier field (29 bits for CAN 2.0B, extended format), remote transmission request (RTR) bit and 4 bits data length code (DLC).

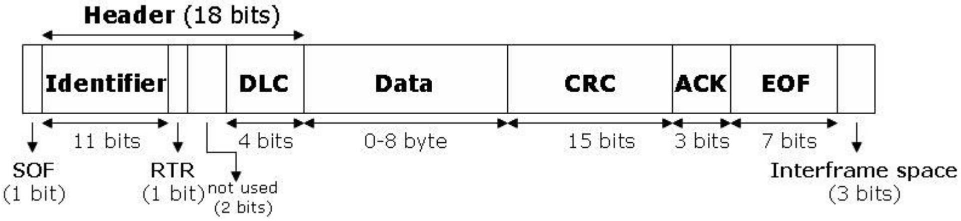


Figure 2.3 CAN 2.0A message format

Identifier field is the label of a CAN frame. Each CAN frame (message) in the bus network should have unique identifier to avoid collisions during transmission. RTR bit specifies the message type in a way that; if it is 0 the message is a data request frame, otherwise it is usual data frame. DLC part carries the information of data field length in bytes. CRC field provides the check of data integrity whether there is any error occurred in frame during bit stuffing or transmission. ACK field contains an ACK slot bit and ACK delimiter bit. During ACK slot the transmitter node is informed whether at least one station received the frame without any error. EOF and intermission frame space fields provide necessary time interval between consecutive message frames. CAN protocol uses bit stuffing with 5-bit length (Non-Return-to-Zero (NRC) encoding). It is the procedure of inserting a bit with different value (0 or 1) into the frame in case five consecutive bits with same value exists in the message. Size of a frame can be calculated including stuffing bits with following equation [5],

$$lm_m = 47 + 8d_m + \left\lfloor \frac{34 + 8d_m}{4} \right\rfloor, \tag{2.1}$$

In Equation (2.1), lm_m is the length of a message in bits, d_m is the number of bytes of data field. In the equation, 47 bits include the overhead and the expression with floor operator ($\lfloor\ \rfloor$) denotes the maximum amount bits required for bit stuffing. Using Equation (2.1), maximum size of a CAN frame, carrying 8 bytes data, can be calculated as 135 bits.

Remote frame enables a node to request a particular message from another node in the network. Remote frame can be used for diagnostics purposes to examine whether nodes, forming embedded network, function properly [21].

Error frame, also called as error flag, is sent by each node in the network that detects error in a message. Error flag is in the form of six successive dominant bits as not obeying the stuffing rule, so that all other nodes are informed about the error by this way.

Overload frame provides capability for a CAN node to require a time interval to get ready for a proper message reception.

2.2.2 CAN Arbitration

CAN bus operates as an AND operator, that means that when more than one node enter a bit to the bus, AND logic operation is applied, and finally the result is observed on the bus. Because of this, on CAN “0” is the dominant, “1” is the recessive bit. CAN arbitration is based on CSMA/CD mechanism to prevent frame collisions during transmission on the bus. At this point, identifier field, belonging to header of a CAN frame and unique for each message, possess an important role for arbitration. As stated previously, each CAN node monitors the bus and when the node detects that the bus is idle, it starts transmission beginning with the identifier field of message. Yet, it is possible that other nodes in the network may start transmission at the same time and only one node would continue sending message. Winner node that will complete transmission without any pre-emption is decided based on CAN arbitration procedure that lasts for the length of identifier field. When a node, monitoring the bus, detects a signal with the same polarity as it has sent, it goes on transmitting message; otherwise it immediately stops transmission and waits for another idle period of bus. The node that monitors bits on the bus with the same polarity of the identifier bits of the message, it is sending, wins the arbitration. Since “0” is the dominant bit on the bus, the message with the identifier field that is the least in value is granted to be transmitted while other ready messages have to wait. On CAN bus once a message wins arbitration, pre-emption of the ongoing transmission is not allowed. Figure 2.4 illustrates CAN arbitration with an example of three nodes’ starting message transmission at the same time. As seen, *Node 1* loses arbitration at the second bit with having recessive bit 1 and stops transmission of *message A*. Then *Node 2* loses at the fourth bit and stops. *Node 3* is the winner node with having the smallest numerical identifier value. Also, figure shows the observed signal on the bus to some extent that is same as *message C*.

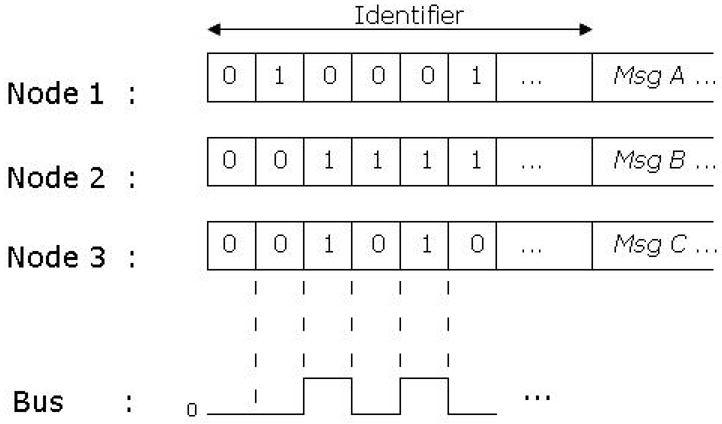


Figure 2.4 An example relating CAN arbitration between three nodes

Arbitration phase lasts for 11-bit time that is the length of identifier field, and since each message in the network has unique identifier, frame collisions are prevented. After each frame transmission, receiver nodes calculate CRC value and compare it with the sent one to observe whether there occurs an error in the message during bit stuffing or transmission. If CRC values are equal, the message is evaluated as valid at the receiver end; otherwise receiver node sends an error flag (six consecutive dominant bits) to inform other nodes about the error. The erroneous message automatically re-enter the next arbitration. In [27], approximate error recovery time is given as between 17 and 31 bit times.

Apart from error detection and correction, CAN also has fault confinement mechanisms. Each CAN node has its own error counter to increase and decrease based on particular events such as, causing a corrupted frame, successful transmission etc. If the error count of a node exceeds the limit value of 256, the node enters bus-off state, in which the controller stops transmitting and receiving messages. By this way, the node confines itself not to block the bus with corrupted frames, unnecessary retransmissions and error flags. Only the host processor in the node or detection of 128 frames with 11 recessive bits can reset the error counter and make the node return back to its initial state (bus-on state) [28].

2.2.3 CAN Priority Scheduling

CAN priority scheduling is performed with scheduling policies that are used to designate the identifier field so to assign priority to a message for transmission. Scheduling policies that can be applied over CAN network can be classified in two groups: fixed (static) and dynamic algorithms.

2.2.3.1 Fixed (Static) Priority Scheduling

In fixed priority scheduling, identifier of messages are designated according to periods (Rate Monotonic, RM) or deadlines (Deadline Monotonic, DM). Priority designation is performed offline (before system run) and identifiers of messages do not change during arbitration phases. References [29] and [30] explains fixed priority scheduling including schedulability analysis of tasks. On the other hand, [31-33] discuss fixed priority scheduling of messages on CAN bus and analyze worst case message response times with providing schedulability requirement as

$$R_m \leq dm_m, \tag{2.2}$$

where R_m is the message response time and dm_m is the deadline of message *m*. Also R_m can be expressed as [31]

$$R_m = tm_m + Q_m, \tag{2.3}$$

where Q_m is the total queuing delay before the transmission of message *m* and tm_m is the transmission time that is the duration to physically transmit message *m* on the CAN bus. Q_m includes delays caused by the blockage of a longest time lower priority message transmission, higher priority message transmissions and queuing jitter that is defined as the delay on queuing of the message in the node. In [34], also schedulability analysis of CAN messages with fixed priorities is discussed including error models and it is shown that existing worst case response time analysis is optimistic especially under high network loads. Since they are simple and there is no need of priority update during arbitration phases, fixed priority scheduling algorithms have advantage of possessing low computational overhead for host processors in nodes.

2.2.3.2 Dynamic Priority Scheduling

Despite the advantages of fixed priority policies, in [35-38] it is shown that dynamic scheduling algorithms perform better by resulting greater percentage of schedulable message sets especially under high network loads. Earliest Deadline First (EDF) algorithm is the main representation of dynamic scheduling policies. Yet, because of having extremely high computational overhead and limited number of identifier bits, approximated EDF scheduling algorithms are applied over CAN. The studies in references [35-38] are basically on the application of EDF over CAN. In [35] and [36] an approximated version of EDF, which is called mixed traffic scheduler, is discussed. MTS approach combines both static and dynamic policies with classifying system messages in three groups: hard deadline periodic messages, hard deadline sporadic messages and non real-time aperiodic messages. Three message groups are differentiated between each other by the first two bits of the identifier. Dynamic scheduling with approximated EDF is applied only on hard deadline periodic messages because of their strict timing requirements, whereas other messages are scheduled using DM scheduling algorithm. EDF approximation is succeeded by quantizing the time axis into equal sized regions called time epochs, and encoding message deadlines according to which region they fall in. In identifier of periodic messages, five bits are reserved for dynamic deadline update, while other later five bits are remained fixed to provide identifier uniqueness.

Moreover, in [37] and [38] Di Natale et al. discusses on an approximated EDF with logarithmic deadline encoding. They define the problem of the complexity of relative deadline computation in addition to having limited number of identifier bits (11-bit) to encode deadlines with high sensitivity. Also, they discuss priority inversion (transmission of lower priority messages first because of imprecise deadline encoding) and analyze its additional delay cost on the worst case response times. They propose to divide the time axis into logarithmic scale, exponentially increasing regions instead of equal sized ones, to encode relative deadlines of messages.

2.2.4 Drawbacks of CAN

CAN networks acquire significant advantages owing to event-triggered behavior such as, flexibility, efficient bandwidth utilization and easy extensibility. Also, error detection/correction and fault confinement mechanisms provide dependability to some extent. Yet, is it enough for safety critical x-by-wire applications? The main disadvantages of event-triggered communication were mentioned before in Table 2.1, which are also valid for CAN. More specifically, event-triggered behavior introduces important drawbacks to CAN such as, difficulty in fault detection (i.e. a faulty node) and having unverified delay bound under worst case requirements. CAN protocol has also drawbacks considering fault detection and fault confinement mechanisms. Automatic retransmission of messages following the error flags in the case of corrupted frame detection makes the bus busy and so induces transmission delay for other messages in the system. In addition, CAN has "*babbling idiot*" problem [4][13], in which a faulty node repeatedly sends high priority messages, blocking the bus. In such cases, the node has to diagnose itself, but this may result in non-detection of faults especially caused by logical errors. Thus, additional fault detection and confinement mechanisms are required to make CAN more dependable and robust, which are necessary for x-by-wire applications.

On the other hand, TTCAN is a hybrid communication protocol enabling transmission of both event and time-triggered traffic. Drawbacks of CAN discussed are mostly solved with TTCAN that is based on TDMA bandwidth allocation scheme with reserving special time windows (time-triggered windows) for hard real-time messages, which makes the protocol more predictable and dependable. Especially under worst case requirements, TTCAN protocol uses the resources efficiently providing bounded response time. Since, nodes are known when to start transmission, error and faulty node detection is much easier in TTCAN. Automatic retransmission of messages upon error occurrence and "babbling idiot" are prevented during time-triggered windows in TTCAN. Moreover, it is easy to adapt CAN controllers to TTCAN implying that migration from CAN to TTCAN is not a costly process [39]. Advantages of TTCAN protocol make it a good candidate for x-by-wire applications.

2.3 Time-Triggered Controller Area Network (TTCAN)

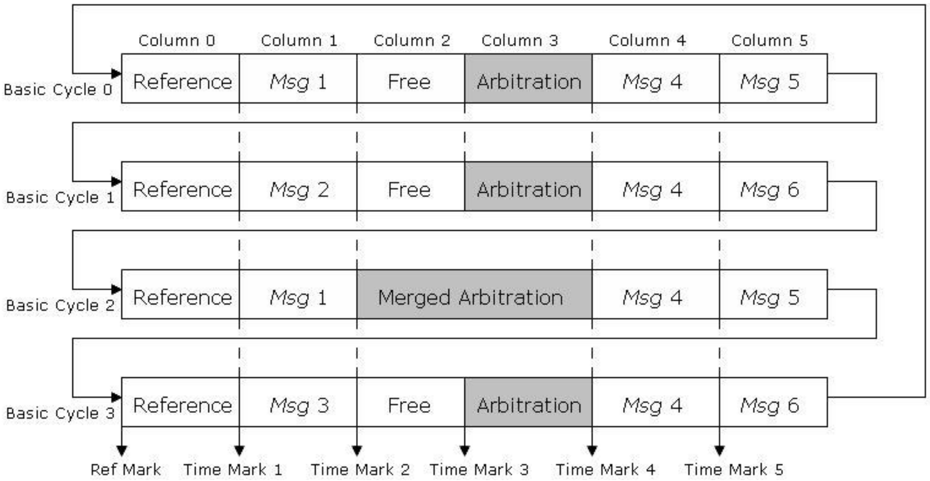
TTCAN communication protocol [25][26][40-42] was developed as a time-triggered version of CAN by Robert Bosch GmbH. TTCAN is implemented as an additional layer on CAN physical and data link layer (DLL). It uses same standards and message format of CAN.

TTCAN is a TDMA based, time synchronous and cyclic bus protocol, which has slots reserved for particular message transmissions. Different from CAN, TTCAN network has a master node that provides time synchronization among nodes by sending a periodic *reference message* that which establishes the cycle-based operation. Also, master node has the capability to change TTCAN operation mode to standard CAN mode (mode change property) [4]. Each node in a TTCAN network has its own local clock that works in *network time unit* (NTU) and time synchronization between these nodes is crucial for time-triggered scheduling operation. TTCAN time synchronization can be implemented in two ways: level 1 and level 2 that is the extension of level 1 [26]. Level 1 synchronization satisfies minimum necessary requirements for time-triggered communication scheduling. In level 1 reference message contains information about the mode of next basic cycle (whether there will be a mode change to standard CAN) and the basic cycle count. On the other hand, level 2 synchronization provides global time information (from the clock of the master node) with high precision (in 2 bytes) in addition to the information provided by level 1. In this thesis level 2 synchronization is assumed to be in use.

2.3.1 TTCAN Communication Scheduling: System Matrix

The communication in TTCAN is based on pre-computed and fixed schedule called TTCAN *System Matrix* (SM) that repeats cyclically during system run. In TTCAN all transmit and receive operations are performed based on the schedule in the SM. SM has a column oriented structure and it consists rows and columns, which form time windows. Each row in the SM is called a *basic cycle* that follow each other. Basic cycles consist of time windows for both time- and event-triggered traffic. Figure 2.3 shows an example of a SM. As depicted in the figure, for each column *j*, the column width (C_j) in "*Basic Cycle 0*" is same for other basic cycles owing to column oriented structure of SM.

A SM may contain four types of windows: reference, exclusive, arbitration and free windows. The first column of SM is assigned to reference windows, during which reference messages are sent at the beginning of each basic cycle by the master node of the network. Reference message that is periodic with the period of basic cycle possess the global time information in addition to some control data. Exclusive windows are reserved for a single particular message (generally periodic) that no transmission of other messages is allowed. In this thesis, exclusive windows are considered only for hard real-time periodic messages. Arbitration windows are time periods where the messages are transmitted / received based on standard CAN arbitration. Due to the event-triggered CAN operation sporadic messages are mostly transmitted in the arbitration windows. It is possible to implement different scheduling policies in arbitration windows that are explained in Section 2.2.3. Finally, free windows are empty windows with no transmission. They are especially reserved for future use such as, extension of a system by adding new nodes and so messages to the TTCAN network.



	Column 0	Column 1	Column 2	Column 3	Column 4	Column 5
Basic Cycle 0	Reference	*Msg* 1	Free	Arbitration	*Msg* 4	*Msg* 5
Basic Cycle 1	Reference	*Msg* 2	Free	Arbitration	*Msg* 4	*Msg* 6
Basic Cycle 2	Reference	*Msg* 1	Merged Arbitration		*Msg* 4	*Msg* 5
Basic Cycle 3	Reference	*Msg* 3	Free	Arbitration	*Msg* 4	*Msg* 6

Figure 2.5 TTCAN system matrix

As illustrated in Figure 2.5, time marks determine the beginning times of reference, arbitration and exclusive windows; in other words, they define the start of transmit and receive operations. Time mark information of messages and arbitration windows is stored in the nodes by registers that are called *triggers*. There are two types of triggers: transmit (*Tx_trigger*) and receive (*Rx_trigger*) triggers. When the local clock in a node reaches to a time mark, transmission or reception process starts with the activation of the trigger pointed to corresponding message. There exist both receive and transmit triggers assigned for exclusive windows and reference windows, but only transmit triggers are defined for arbitration windows. In time windows, once a message is triggered, the related Tx_trigger stays active by triggering the message transmission for a *Tx_Enable* period that is the maximum time interval during which the transmission should start. If it does not start, the transmission of the message is not allowed during the remaining time of the window. By this way, any possible overrun of a message transmission into the next window is prevented with satisfying temporal isolation between different traffics (time and event triggered) and time windows.

For each time-triggered message that a node transmits or receives, there exists at least one transmit or receive trigger pointing to the related message in the corresponding node. Triggers are described with three parameters; the column number ($Column_j$), cycle offset (CO) and repeat factor (RF). Cycle offset is the number of the first basic cycle in which the message must be transmitted or received. Repeat factor determines the period of time windows (exclusive or arbitration) in terms of basic cycle. If a time window does not appear periodically in SM, the repeat factor value is assigned to zero [26]. In a TTCAN network, a node only needs to know about the messages that it transmits or receives in the system for the sake of effective memory utilization. In other words, nodes possess time mark information only about their respective messages that they are expected to transmit or receive.

Table 2.2 An example relating the trigger and window properties of a TTCAN node based on the SM in Figure 2.5

	Window Type	**Time Mark *id.***	**Trigger Type**	**$Column_j$**	**CO**	**RF**
Msg_{Ref}	Reference	0 (Ref Mark)	Rx_Trigger	0	0	1
Msg 1	Exclusive	1	Tx_Trigger	1	0	2
Msg 2	Exclusive	1	Rx_Trigger	1	1	0
$Arb_{0,3}$	Arbitration	3	Tx_Trigger	3	0	1
$Arb_{2,2}$	Arbitration	2	Tx_Trigger	2	2	0

For example, let us take a standard TTCAN node under consideration for the SM given by Figure 2.5 and assume that the node is expected to transmit Message 1 (*Msg* 1) and receive Message 2 (*Msg* 2). The related window and trigger properties of the node are given in Table 2.2. $Arb_{0,3}$ and $Arb_{2,2}$ in the table denotes the arbitration windows with the index of basic cycle and column numbers respectively. Different from other nodes in the network, the master node possesses Tx_Trigger for the reference message.

2.3.2 Related Work on TTCAN Communication Scheduling

Communication between nodes in a TTCAN network is managed according to TTCAN system matrix that is a predefined and static schedule. So, design of SM structure is crucial for real-time performance of an embedded system. There are several studies [43-47] in literature on strategies for SM construction. In these studies some performance metrics are defined to obtain a basis for the development of scheduling approaches. Also, proposed approaches for SM construction is tested and evaluated with respect to defined performance metrics over benchmark message sets such as Society of Automotive Engineers (SAE) [6] and Peugeot-Citroen Automobiles Company (PSA) [48] benchmark sets as well as artificial, example message sets.

In [43] and [44] a stochastic optimization algorithm is proposed. Jitter, appointed as a performance metric, is defined as the proportional amount of time shift between the message transmission time instants and the period, and it is aimed to obtain optimum solution for a minimum jitter. The approach is based on the construction of a high number of initial set of system matrices. Then several transformation techniques (i.e. cell swap, column swap, vertical mirror and horizontal mirror etc.) are applied randomly over the initial set. After sufficient number transformation iterations, optimum SM for minimum jitter is obtained. The approach is applied over PSA benchmark set. In that work, only periodic messages are considered and bandwidth utilization, bandwidth loss due to different message sizes and number of triggers are not discussed.

In [45] and [46] heuristic scheduling concepts are discussed. In [45] Albert et al. focus on two important concepts that are arbitration window placement and basic cycle number/length with respect to number of arbitration triggers, response frequency and bandwidth loss due to reference messages. Based on performed tests and results on a self-constructed message set, they propose that evenly distributed arbitration windows and long basic cycles produce better real-time performance. Although longer basic cycle induces more number of triggers for the nodes, it provides better response frequency and less bandwidth loss due to reference messages. In that work, number of triggers is considered for only arbitration windows and messages are assumed to have equal length as 8 bytes data. Also, number of triggers for periodic messages and network utilization are not discussed. In another study [46], basically real-time schedulability analysis of messages is focused on. The messages in the system are classified in three groups based on their response time requirements: hard, firm and soft real-time messages. At first, total SM duration and basic cycle length are obtained based on two alternative strategies: minimizing the number of basic cycles with increased basic cycle length and minimizing basic cycle length with increasing the number of basic cycles. Exclusive windows are assigned to hard real-time messages; on the other hand, firm real-time messages are directed to arbitration windows. Different system matrix designs system analyzed considering number of triggers and response times.

In [47] a TTCAN scheduler, Smart-PLAN is described. This approach proposes message placement on the SM according to slack time (remained time to deadline after transmission) of messages. Since message instances are placed on the SM in a row oriented manner, column widths are determined during the placement on the first basic cycle. The method is applied on SAE benchmark set and the results are compared with different scheduling techniques with respect to bus utilization. In that work mostly message schedulability and bus utilization are discussed whereas jitter, sporadic messages and number of triggers are not presented.

CHAPTER 3

TTCAN COMMUNICATION SCHEDULING COMPONENTS

In Chapter 3, necessary components that have important roles in TTCAN communication scheduling will be discussed in addition to providing the problem statement. By this way it is aimed to designate the borders of both workspace and solution frame of SM design. These components are defined as; assumptions, protocol and hardware constraints, performance metrics and message set properties. The first section of the chapter gives background information on SM design components including assumptions, message set properties and performance metrics adopted from references. The following chapters relate the protocol constraints and additional performance metrics. In this thesis, all these components are considered during SM design, analysis and performance evaluation.

3.1 Background Information

In this section background information about message set properties and some of the performance metrics are given, which are adopted from [5]. The notation and assumptions in [5] are also used in the thesis and explained under the section of Assumptions.

3.1.1 Assumptions

The following titles explain the assumptions, related with message properties and TTCAN communication protocol, which are used throughout the thesis. By defining the assumptions, it is aimed to describe the frame of the proposed solutions for TTCAN scheduling in a clearer and easier manner.

Message format: CAN 2.0A message format that has 11-bit identifier field is used throughout application examples and tests in the thesis. Size of a message is calculated with Equation (2.1) and this equation can be written alternatively as following equation in terms of data bits (b_m) in a message frame

$$lm_m = 47 + 8\lceil b_m/8 \rceil + \left\lfloor \frac{34 + 8\lceil b_m/8 \rceil}{4} \right\rfloor, \tag{3.1}$$

where the term with floor operator is for the maximum number of bits required for bit stuffing.

Reference message: Level 2 time synchronization is assumed to be applied, which makes data field of the reference message 4 bytes. By using Equation (3.1), maximum size of the reference message ($l_{\mathrm{Re}f}$) is calculated as 95 bits.

Network Time Unit (NTU): Network time unit, as the unit of cycle time in the TTCAN network, is assumed to be equal to bit-time (τ_{bit}) that is the duration of 1 bit of data transmission on the bus [5]. Bit-time value depends on the bus bandwidth (data rate, b_r); i.e. for 1Mb/s bus bandwidth, $\tau_{bit} = 1\mu\sec = 1NTU$ that is obtained with the following equation

$$\tau_{bit} = 1/b_r \ , \tag{3.2}$$

Message periods and deadlines: For a periodic message in the network, the period (pm_m) and deadline (dm_m) are assumed to be equal to each other ($pm_m = dm_m$). On the other hand, for a sporadic message, minimum interarrival time (*mit*) is assumed to be equal or greater than the deadline ($pm_m \geq dm_m$). Moreover, only real-time messages are considered in TTCAN communication scheduling and transmitted messages are assumed to be received by all nodes (except the sender) in the network.

Tx_Enable period: Tx_Enable period, defined in Section 2.2.1, is taken as $16\tau_{bit}$ time [5]. Thus, *transmission time* of a message (necessary time to transmit the message physically on the bus) may be expressed including Tx_Enable period as

$$tm_m = (lm_m + 16) \cdot \tau_{bit} \, , \tag{3.3}$$

3.1.2 Message Set Properties

A message set *M* consists of two types of real-time messages: periodic (M_P) and sporadic (non-periodic) messages (M_S). They may be also called as time-triggered and even-triggered messages respectively. Periodic messages are considered as hard real-time messages that demand bounded response delay with small jitter and dependability and generally exclusive windows are assigned to them. Periodic messages are described with three attributes: period (pm_m), deadline (dm_m) and message length in bits (lm_m). On the other hand, sporadic messages are described with minimum inter-arrival time (mit_m) (may be seen as period as well), deadline and message length and they are generally transmitted during arbitration windows.

The periodic message subset of *M* can be shown as $M_P = \{M_1, M_2, ..., M_G\}$. In this set M_1 is defined as the periodic message with minimum period pm_1 and the index *G* denotes the number of messages in the message set M_P, where M_G is the last message with the maximum period. The messages in the set M_P are assumed to be ordered according to increasing value of periods. On the other hand, the subset of sporadic messages is denoted with M_S. Sporadic messages are ordered in the subset (M_S) according to their deadlines.

In this thesis, based on provided background information message sets are classified under two groups according to period properties that the periodic message subset possesses, which has an important role on SM design. These message set groups are defines as *ideal message set* and *non-ideal message set.*

Ideal message set: Ideal message sets have an important property considering the period values of messages [5]. For a periodic message set $M_P = \{M_1, M_2, ..., M_G\}$, if periods of messages, other than M_1, are in the form of $2^j\, pm_1$, $j \le j_{\max} \le 6$ where all j values are integer and $j_{\max}$ is the power of the maximum period message, M_G, with the period value of $2^{j_{\max}}\, pm_1$, then the message set is described as *ideal*. As stated earlier in Section 3.2, the number of lines (*L*) constituting the SM should be power of 2 in TTCAN protocol, so that having an ideal message set does not only provides easiness in SM construction but also minimum trigger count and jitter achievement for periodic messages. Ideal message set property is explained in more detail with an example in the following chapter under title of periodic message scheduling. For example, PSA benchmark [44][48] is an ideal message set consisting of 12 periodic messages exchanged between 6 nodes.

Table 3.1 PSA benchmark periodic message set

Message ***id.***	**Length** (bits)	**Period** (μsec)	**Deadline** (μsec)	**Transmitter Node** (Node *id.*)
1	135	10000	10000	Engine controller (1)
2	85	10000	10000	Wheel angle sensor (2)
3	85	20000	20000	Engine controller (1)
4	75	10000	10000	AGB (3)
5	105	20000	20000	ABS (4)
6	105	40000	40000	ABS (4)
7	95	10000	10000	ABS (4)
8	105	40000	40000	Bodywork sensor (6)
9	95	20000	20000	Device y (5)
10	125	80000	80000	Engine controller (1)
11	105	40000	40000	AGB (3)
12	65	80000	80000	ABS (4)

Among the messages given in Table 3.1, M_1 has the minimum period that can be denoted as pm_1. As seen from the table, periods of messages can be written in the form of $2^j pm_1$, that is; messages M_2, M_4 and M_7 have period of $2^0 pm_1$, M_3, M_5 and M_9 have period of $2^1 pm_1$, M_6, M_8 and M_{11} have period of $2^2 pm_1$, and finally M_{10} and M_{12} have period of $2^3 pm_1$.

Non-Ideal message set: A message set, in which message periods do not possess above mentioned is defined as non-ideal message set in the thesis. The approach for scheduling such message sets is also explained in the following chapter.

Another important concept, also discussed in [5], is the schedulability of message sets, which is the requirement of the transmission of messages before their deadlines. A message set is identified as schedulable if the necessary time to transmit the total number of bits is equal or smaller than the total SM duration. In other words, if the total duration of necessary SM time windows, during which periodic and sporadic messages are transmitted, exceeds the total SM duration then the message set cannot be seen as schedulable.

3.1.3 Performance Metrics

Network utilization (NU): Network utilization relates how efficiently bandwidth is utilized by the SM in the scope of time-triggered traffic [5] and it can be expressed as

$$NU = \frac{\sum_{M_P} D_m}{\left(\sum_{M_P} A_m\right) + L \cdot A_{\mathrm{Re}\,f}}, \tag{3.4}$$

where D_m denotes the sum of the time in the SM used for transmitting data of a periodic message m that belong to set M_P, and $\sum_{M_P} D_m$ is the total transmission time of data bits of messages in M_P. Similarly, A_m denotes the sum of exclusive window durations in which periodic message m is transmitted and $\sum_{M_P} A_m$ is the total amount time of the message transmission windows (exclusive windows) in the SM for the message set M_P. Also, the expression $L \cdot A_{\mathrm{Re}\,f}$ is the total time allocated to reference message transmission in the SM.

Matrix load (ML): Matrix load represents the load of the SM. In other words, *ML* gives an intuition about the capacity of the SM to provide schedulability for new signals and nodes in the TTCAN network [5]. It can be expressed as

$$ML = \frac{\left(\sum_{M_P} A_m\right) + L \cdot A_{\mathrm{Re}f}}{T}, \tag{3.5}$$

where T denotes the total system matrix duration with the expression of

$$T = B \times L, \tag{3.6}$$

Jitter (J): Jitter relates the amount of time shift between period and the consecutive transmission instances of a periodic message [5][43]. In other words, it is the proportion of total response delays of the message instances in the SM to the SM duration (T). Response delay (transmission delay, D_m) of a message can be calculated as,

$$D_m = \sum_i \left| e_{m,i} - a_{m,i} \right|, \tag{3.7}$$

where i represents the transmission instances (starting time instants of exclusive windows) of message *m* in the SM and $e_{m,i}$ is the expected beginning time for transmission whereas $a_{m,i}$ is the actual beginning time of transmission. The expected beginning times of a periodic message can be defined as arrival times of message instances. The time instant at which the message appears at first in the SM is taken as the arrival time of the first message instance, and so following expected beginning times for transmission can be obtained by adding the respective message period to first arrival time of the message. By using D_m value, jitter of message *m* can calculated as,

$$J_m = \frac{D_m}{T}, \tag{3.8}$$

So total system jitter for message set M_P can be expressed as

$$J = \frac{1}{T}\sum_{M_P} D_m, \tag{3.9}$$

Number of triggers (T_N): The constraint on the trigger count per node is given in Equation (3.12). This condition also constraints the number of messages and nodes in the system with making the system extension and evolution (adding new nodes and messages to TTCAN network) difficult. Thus, number of triggers can be defined as a performance metric that, forcing the use of small number triggers to the system is a good property for a SM [5].

3.2 Protocol and Hardware Constraints

The constraints stipulated by TTCAN protocol and communication controller chip perform an important role on SM design. In other words, the constraints define the limits for the work space of SM construction. Following titles explain these constraints.

Number of basic cycles (L): Number of basic cycles (lines) in a SM should be a power of 2. Also, *L* is limited to the maximum value of 64. This condition can be expressed in Equation (3.10).

$$L = 2^q,\ 0 \le q \le 6 \tag{3.10}$$

Basic cycle duration (B): Maximum allowed basic cycle duration is constrained to the value of $2^{16}\ NTU$.

$$B \leq 2^{16} NTU \ , \tag{3.11}$$

Number of triggers: The maximum number of triggers in each node in a TTCAN network is limited to the value of 32 and this can be expressed as

$$T_N = Tx_N + Rx_N \leq 32 \, , \tag{3.12}$$

where Tx_N and Rx_N denote the number of transmit and receive triggers respectively, while T_N represents the total number of triggers of node *N*.

Bus bandwidth (b_r): The bus bandwidth, ranging from 125 kb/s to 1 Mb/s, for high speed CAN is also valid for TTCAN protocol.

3.3 Additional Performance Metrics

Performance metrics provide the necessary platform of evaluation criteria for both analysis and design of system matrices. Real-time communication performance requirements of a networked system are described in terms of these metrics according to which TTCAN SM is constructed. The following performance metric, bandwidth loss is proposed in this thesis in addition to ones adopted from [5] and [43], which can be used for real-time performance evaluation of the SM.

Bandwidth loss (Bw_{Loss}): Bw_{Loss} is used as a performance metric to relate the total amount of time (in $\mu\sec$) per matrix cycle spent for no transmission in exclusive windows in addition to transmission of reference messages. Three bandwidth loss types for periodic messages are defined in the thesis: *reference message loss* (Bw_{loss}^{ref}), *unused window loss* (Bw_{loss}^{uw}) and *in-window loss* (Bw_{loss}^{iw}). Bw_{loss}^{ref} relate the amount of bandwidth loss per matrix cycle due to transmission of reference messages. Bw_{loss}^{uw} occurs due to reserved but unused exclusive windows in the SM and Bw_{loss}^{iw} is caused by unequal sizes of messages, which are placed on the same column of the SM. The values of all bandwidth loss types are measured in per matrix cycle (for the duration of *T*). These loss types will be detailed in the next chapter. Bw_{Loss}, denoting the total bandwidth loss per matrix cycle, is highly related to the performance metric, *NU*. A_m value in *NU* includes bandwidth loss caused by unused windows and in-window unused time so, A_m can be expressed by decomposing these components

$$A_m = F_m + Bw_{loss,m}^{uw} + Bw_{loss,m}^{iw}, \tag{3.13}$$

where F_m, as expressed in Equation (3.14), denotes the total transmission time in the exclusive windows that are assigned to message *m* in the SM.

$$F_m = \sum_e (lm_m + 16) \cdot \tau_{bit}, \tag{3.14}$$

where *e* denotes the set of time instances for exclusive windows assigned to related message *m*, during which transmission occurs.

Total bandwidth loss is the actual sum of for all messages is defines as

$$Bw_{Loss} = Bw_{loss}^{ref} + Bw_{loss}^{uw} + Bw_{loss}^{iw}, \tag{3.15}$$

where Bw_{loss}^{ref} can be expressed as

$$Bw_{loss}^{ref} = L \cdot A_{\mathrm{Re}\,f} \, , \tag{3.16}$$

so, Equations (3.4) and (3.5) can be rewritten in terms of bandwidth loss as

$$NU = \frac{\sum_{M_P} D_m}{\left(\sum_{M_P} F_m \right) + Bw_{Loss}} \, , \tag{3.17}$$

$$ML = \frac{\left(\sum_{M_P} F_m \right) + Bw_{Loss}}{T} \, , \tag{3.18}$$

which means *NU* value decreases whereas, *ML* increases with the increasing bandwidth loss in the SM.

Slack time: Slack time is the difference between the deadline and the time instant of transmission completion ($tc_{m,i}$) of a message. This performance metric gives an idea about the schedulability degree of a message set with the respective SM. In the thesis it is especially used for schedulability analysis of sporadic messages. Slack time of a message for the i^{th} instance can be expressed as

$$slack\ time_{m,i} = dm_{m,i} - tc_{m,i} \, , \tag{3.19}$$

As seen from the equation, in case of a deadline miss, slack time value becomes negative. Thus, higher total slack time value of messages means higher robust and dependable network owing to the structure of SM.

Bus Utilization (BU): It relates the proportion of the bus busy time by message transmissions including sporadic messages to total time of system run. *BU* can be expressed as

$$BU = \frac{bus_{busy}}{T_R}, \tag{3.20}$$

CHAPTER 4

TTCAN MESSAGE SCHEDULING: SYSTEM MATRIX DESIGN

In Chapter 4, message scheduling in TTCAN communication protocol will be discussed, which basically depends on the structure of the SM. As stated before, this thesis constitutes an extension of the work on a systematic approach for SM design proposed in [5]. In the first section, the related studies on scheduling of both ideal and non-ideal periodic message sets in [5] will be summarized, also providing comments and related contributions of the thesis. In this chapter, the proposed and adopted methods in SM design are analyzed with the help of TTCAN scheduler tool, also to be described, to observe their effects on defined performance metrics. Chapter 4 can be divided into two main parts as SM design for periodic messages and sporadic messages. The first part explains SM construction for ideal periodic message sets with mainly focusing on scheduling for message sets with unequal message sizes. Also, for non-ideal message set scheduling the effects of methods for scheduling are analyzed theoretically and evaluated by applying on example message sets. The second part explains the proposed solution for sporadic message scheduling.

4.1 Problem Formulation and Contribution of the Thesis

As stated in Section 2.3.1 in the background chapter, communication in TTCAN is performed based on a scheduling matrix that is computed offline and fixed during system run. The TTCAN scheduling matrix consists of rows (basic cycles) and columns that form time windows, during which messages are transmitted based on the schedule of the matrix. As being a column oriented structure, column widths of the first row are same for the other rows, which are defined according to message transmission times.

There are certain issues related to the TTCAN scheduling matrix construction because of performance requirements and the operation of TTCAN as described in the standard, which are addressed in [5]. These problems can be listed as follows:

- The messages must be sent before their respective deadlines with transmitting periodic ones according to their respective periods.
- Efficient bandwidth utilization that implies minimum bandwidth loss in time windows (reserved but not used time durations in the matrix) as well as minimum message load in the scheduling matrix with the aim of increasing network utilization and improving system extensibility, which is the feasibility of a system for the addition of new messages and new nodes.
- Minimum number of triggers for both satisfying protocol constraint and improving system extensibility.
- Satisfying protocol constraints and standards while providing better communication performance as much as possible.

In [5] an analytic approach is followed to find optimal scheduling matrices. The approach in [5] firstly analyzes message properties and results that messages have to be transmitted with periods that are powers of 2 (that is called ideal message set property in the thesis as explained in Section 3.1.2) and it is exploited to simplify the solutions, in which the aim is to investigate optimal scheduling strategies such as, zero jitter, minimum number of triggers and maximum network utilization with giving a formal description and including the defined performance metrics.

The contributions of this thesis can be listed as follows:

- Three bandwidth loss types are defined as performance metrics and they are related to adopted metrics to enhance the definition of efficient bandwidth utilization.
- A method is developed and implemented for an approach provided in [5] for scheduling of periodic messages that have ideal property, which proposes solution to the optimization problem
- Analyses are performed on the formal methods given in [5] for scheduling of periodic messages that do not have ideal property, and application examples are provided to relate the effects of formal methods on defined performance metrics with the help of scheduler tool that will described in Section 4.3 in more detail.
- A new scheduling approach is proposed for the sporadic messages that is different than the approach presented in [5] that guarantees that the messages are transmitted before their deadlines. Also, tests are performed to evaluate scheduling matrices in the scope of better sporadic communication performance.

4.2 Background Information on Periodic Message Scheduling

In this section, the related work on periodic message scheduling will be summarized to provide the better understanding for the following sections. In this scope, firstly scheduling of ideal periodic message set with equal message lengths will be explained, then the approach for unequal message lengths will be provided. Secondly, non-ideal message set scheduling and proposed methods will be discussed. All provided material in this section is adopted from [5].

Ideal message set scheduling is discussed in [5] in detail with the assumption of equal message lengths. The following titles explain the SM design for an ideal periodic message set $M_P = \{M_1, M_2, ..., M_G\}$, in which messages are ordered with increasing period so that the period of the first message, pm_1 is the minimum period while last message period pm_G is the maximum period in the set.

Ideal message set with equal message lengths: As explained previously, ideal periodic message sets are described upon the property of message periods that can be written as in the form of $2^j\, pm_1$. SM design for an ideal message set with equal message lengths is a simple, column-oriented procedure that can be described as follows:

- The minimum message period pm_1 is assigned as the basic cycle duration ($B = pm_1$) and maximum message period as the SM duration ($T = pm_G = 2^{j_{\max}}\, pm_1$), where G denotes the number of messages in the message set M_P. Since $T = B \times L$ from Equation (3.6), by substituting B and T values, number of lines in the SM can be calculated as

$$L = \frac{2^{j_{\max}}\, pm_1}{pm_1} = 2^{j_{\max}}\ , \tag{4.1}$$

- Exclusive windows are assigned to messages starting from messages with smaller periods. Also, corresponding Tx_triggers are introduced for the messages, placed in the SM.
- If there are not enough windows in the column for the message to be placed, the windows in the next free column are assigned and remaining free window(s) in the previous column are assigned to the message with higher period.

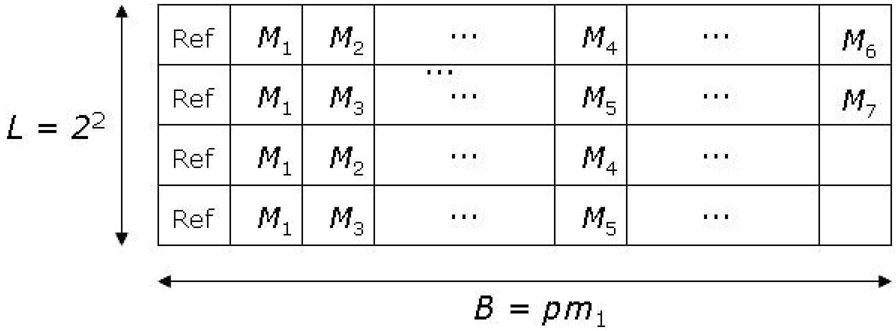
By this procedure number of appearances of a periodic message in the SM can expressed with the following equation as

$$k_m = \frac{T}{pm_m}\ , \tag{4.2}$$

which can be equally written as $nm_m = \frac{2^{j_{\max}}\, pm_1}{2^j\, pm_1} = \frac{2^{j_{\max}}}{2^j}$. So, the total number of periodic message appearances in the SM becomes $M := \sum_{m=1}^{G} \frac{T}{pm_m}$. Thus number columns necessary for periodic message scheduling can be calculated with the equation $\left\lceil \frac{M}{2^{j_{\max}}} \right\rceil$, where $2^{j_{\max}}$ is the number lines of the SM by Equation (4.1).

This procedure can be related with an example of ideal message set with 7 equal length periodic messages. M_1 has the period of pm_1, M_2, M_3, M_4 and M_5 have periods of $2pm_1$ and M_6, M_7 have periods of $2^2\, pm_1$. *B* and *L* values can be determined as $B = pm_1$ and $L = 2^2$. By applying above procedure, the SM for the corresponding message set is constructed as given in Figure 4.1.

Figure 4.1 An example relating SM design for an ideal message set with equal message lengths



Ref	M_1	M_2	⋯	M_4	⋯	M_6
Ref	M_1	M_3	⋯	M_5	⋯	M_7
Ref	M_1	M_2	⋯	M_4	⋯	
Ref	M_1	M_3	⋯	M_5	⋯	

The duration of each assigned column (C_d) is equal because of the equal message sizes. C_d values that are also transmission time of messages because of equal message sizes can be expressed as

$$C_d = tm_m = (l_m + 16) \cdot \tau_{bit},\ 1 \le d \le d_{\max}, \tag{4.3}$$

where $d_{\max}$ is the last assigned column number in the SM.

Since it is an ideal periodic message set, the messages are placed on the SM in an exact accordance with their respective periods. This results in no difference between $e_{m,i}$ and $a_{m,i}$ values, which implies zero jitter for periodic messages. Also, by determining the basic cycle length as minimum period satisfies minimum number of triggers for nodes (one *Tx_trigger* for each message) is satisfied.

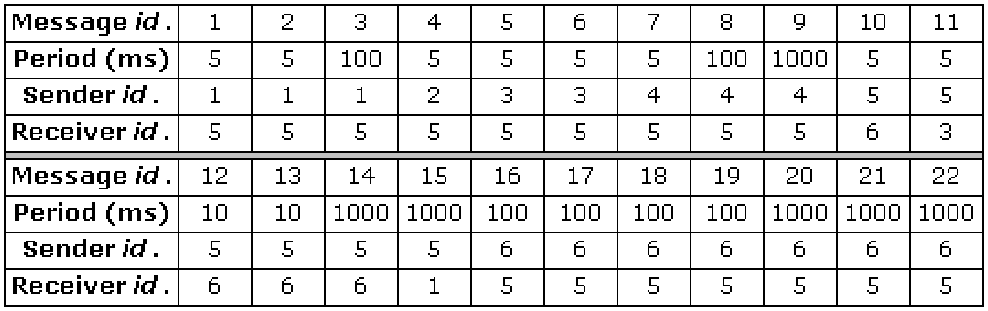
Ideal message set with unequal message lengths: Two solutions are proposed in [5] to construct the SM for this type of message sets.

The first solution is to place the messages onto SM windows as if the messages have equal lengths. This approach is explained in the previous part as SM design for ideal message set with equal message lengths. By this approach, since lengths of messages are not equal, each column width (C_j) is determined as the transmission time of the message with the largest size in the corresponding SM column. As stated before, this generates time durations, during which no transmission occurs, caused by the difference transmission times of the messages placed in the same column, with resulting in a partial bandwidth loss in exclusive windows, which is defined as in-window loss in Section 3.3.

Secondly, a solution is proposed as constructing the SM columns with similar sized messages and this approach yields an optimization problem for maximizing the NU by minimizing the value of the term $\sum_{M_P} A_m$ in Equation (3.4). In the following section, a method is proposed to provide optimum solution for SM design satisfying minimum in-window loss and so maximum *NU*.

Non-ideal message sets: In [5], it is shown with an example of the SAE benchmark periodic message set that additional methods are required for scheduling of a non-ideal message set. SAE periodic set is given in Table 4.1.

Table 4.1 SAE benchmark periodic message set



Message *id*.	1	2	3	4	5	6	7	8	9	10	11
Period (ms)	5	5	100	5	5	5	5	100	1000	5	5
Sender *id*.	1	1	1	2	3	3	4	4	4	5	5
Receiver *id*.	5	5	5	5	5	5	5	5	5	6	3

Message *id*.	12	13	14	15	16	17	18	19	20	21	22
Period (ms)	10	10	1000	1000	100	100	100	100	1000	1000	1000
Sender *id*.	5	5	5	5	6	6	6	6	6	6	6
Receiver *id*.	6	6	6	1	5	5	5	5	5	5	5

As a non-ideal message set, SAE benchmark periodic set [6] consists of 22 messages that are exchanged between 6 nodes. The *id*s of sender and receiver nodes of messages are also given in the table. In SAE message set, all message lengths are equal with obtaining 1 Byte of data, which result in the message size of 65 bits.

In the approach of SM construction of non-ideal message sets discussed in [5], by defining T as *least common multiple* (lcm) of the message periods, it is rewritten in the form of

$$T = pm_m \cdot 2^{j_{\max}-j} \cdot K \,, \tag{4.4}$$

where j is chosen to make the term $2^{j_{\max}-j}$ contain all 2 multiples that K is defined as the *nonharmonic divisor* with containing no multiples of 2. Also, the offset between occurrences of a message in consecutive SM lines is defined as $o_m = \left\lceil \frac{T}{L \cdot pm_m} \right\rceil \cdot pm_m - \frac{T}{L}$ for $pm_m < B$ and $o_m = pm_m - \left\lceil \frac{pm_m \cdot L}{T} \right\rceil \cdot \frac{T}{L}$ for $pm_m > B$. Then by substituting T and L values and doing necessary simplifications, it is concluded that 2^j relates the number of consecutive lines, in which the message is to be scheduled in different columns and *K* value gives the number of Tx_triggers to schedule the corresponding message. This approach is also shown on the SAE periodic message set. At the first step, *T* is defined as *lcm* of message periods, which yields $T = pm_m \cdot 2^{j_{\max}-j} \cdot K = 10^6\ \mu\text{sec}$. Then, *L* value is chosen as $L = 2^{j_{\max}} = 2^5$ that results in $B = \frac{T}{L} = 31250\ \mu\text{sec}$ by Equation (3.6) with satisfying protocol constraints. If M_1 is taken under consideration, Equation (4.4) is written for the message as $T = pm_1 \cdot 2^{j_{\max}-j} \cdot K$ and by substituting the respective values in the equation $10^6 = 5000 \cdot 2^{5-j} \cdot K$, which yields $2^3 \cdot 5^2 = 2^{5-j} \cdot K$ so that $2^j = 4$ and $K = 25$. The obtained results imply that M_1 can be scheduled in different columns in 4 consecutive lines of the SM with 25 Tx_triggers. Applying the same procedure, it also generates 25 Tx_triggers for M_2 and 5 Tx_triggers for M_3 so that total number of only Tx_triggers becomes 55 ($25+25+5$) for *Node* 1, which exceeds the protocol constraint for number of triggers a node can possess as given by Equation (3.12). Thus, to design SM for such message sets requires additional operations that, the methods that are defined as reduction of message periods and reduction of matrix cycle are proposed in [5] for this purpose. By applying these methods, it is aimed to put message periods into the ideal form to design the SM with satisfying protocol constraints. The defined methods are also applied on the SAE message set example and minimum number of triggers and zero jitter are obtained with satisfying protocol constraints.

In Section 4.4.2, the methods for non-ideal message set scheduling are examined in terms of their effects on performance metrics from different aspects. The methods are analyzed theoretically and then evaluated over both a self constructed and SAE periodic message set examples with the help TTCAN scheduler tool that will also explained in Section 4.3. The important role of system performance requirements on SM design is shown by these analyses.

4.3 TTCAN Scheduler Tool

TTCAN scheduler tool is developed with the aim of helping SM construction and analysis. It is developed in the C++ programming language (visual studio 2003). TTCAN scheduler tool can be examined under two topics: scheduling of ideal and non-ideal periodic message sets. For ideal message set scheduling (both with equal and unequal message lengths), the tool uses message properties (period, deadline and size) and allocated total column width for periodic message scheduling ($C_{Periodic}$) as inputs. By this information, it designs the SM with providing maximum *NU* while imposing minimum number of triggers as the implementation method will be explained in Section 4.4. Secondly, the use of the tool on non-ideal periodic message sets consists of communication performance analysis. The tool inputs message properties, reduced message periods and BC duration. The inputs of reduced message periods and BC duration can be provided externally, also the program can define reduced periods and BC duration automatically, by taking *B* as the minimum message period, pm_1 and reducing other message periods to closest value of $2^j pm_1$. Based on input values, the methods for scheduling such message sets, such as modification on BC duration and message periods that will be explained in the following section are analyzed by the tool with providing the results of related performance metrics.

4.4 SM Design for Periodic Message Set

In this section, the discussion on SM design mentioned in the previous part will be extended for both ideal and non-ideal message sets. In the first subsection, ideal message sets with unequal message lengths are considered and a method is proposed as a solution to defined optimization problem mentioned in the previous section. In the second subsection, non-ideal message sets are taken under consideration, and methods for SM construction are analyzed.

4.4.1 SM Design for Ideal Message Set

Different from the previous part, in this section messages in the ideal message set $M_P = \{M_1, M_2, ..., M_G\}$, in which the messages are ordered with increasing message periods, do not necessarily have equal message sizes. Based on the approach of constructing the SM columns with similar sized messages, a method is proposed to provide optimum solution for maximizing the NU .

The proposed method uses the depth first algorithm by investigating all possible SM configurations to provide the optimum one that maximizes NU by minimizing in-window loss (Bw_{loss}^{iw}). The proposed method is also implemented in the TTCAN scheduler tool by using a column oriented, tree based structure. The algorithm uses total column width reserved for periodic message scheduling ($C_{Periodic}$) and message properties (period, deadline and size) as the inputs. Elements of the tree structure are the columns that form the SM. Each tree element, belonging to column class, contains the information of column *id.*, parent column *id.*, *unplaced message list*, *total cost* value and *used column width* (C_{used}). Column *id.* and parent column *id.* denote the identification numbers of the column itself and the parent column that generates it respectively. Unplaced message list contains the list of messages that are not scheduled yet, in other words, the messages that are not placed in any parent columns up to corresponding column (including itself). Total cost value gives the information about the total in-window loss up to corresponding column including itself and total used column width is the total column duration used by parent columns up to corresponding column (including itself). These stored properties of each column are important for implementation of the algorithm. Last tree elements are called the leaf columns that are the final columns of the completed system matrices that contain all messages scheduled with placing in the columns. The leaf columns obtain the information of total cost that is necessary to find optimum

SM with minimum in-window loss. By this purpose all generated leaf columns, belonging to appropriate system matrices that contain all messages placed with not exceeding the allocated column width ($C_{Periodic}$), are added to *leaf column array*. The workflow of the procedure starts with the generation of the first element which is the reference message column. This element is the parent of all other consecutive columns to be generated in the tree with possessing no parent column *id*. As explained previously, also this column obtains its own peculiar information of column *id*., unplaced message list and total cost etc. To construct the tree structure, at first the reference message column is taken under consideration. The first message (with the smallest period) in its unplaced message list is chosen and this message appears as being placed in all columns generated from the parent column. In other words, all possible column elements are generated with different message placement configurations but with all having the message that has been chosen at first. The generation of a column element depends on a criteria that, if the total used column width is larger than $C_{Periodic}$, the column under consideration is not created. The same column generation procedure is applied also by newly generated columns, this process lasts until there is no unplaced message left for each column in the tree.

After construction of the tree by generation process, the leaf column array is investigated to find the optimum SM configuration. If leaf column with minimum total cost is found, the respective SM is formed by combining all parent columns up to leaf element with back tracing the parent column *id*s, stored in each column forming the SM, starting from the leaf column with finally coming to reference column.

Figure 4.2 illustrates the approach with *n-level* tree structure, which implies that there exist *n* columns in the SM allocated for periodic message scheduling; yet note that this does not mean all configured system matrices possess *n* columns for periodic message scheduling. It may be possible to obtain system matrices that schedule periodic messages using less number of columns.

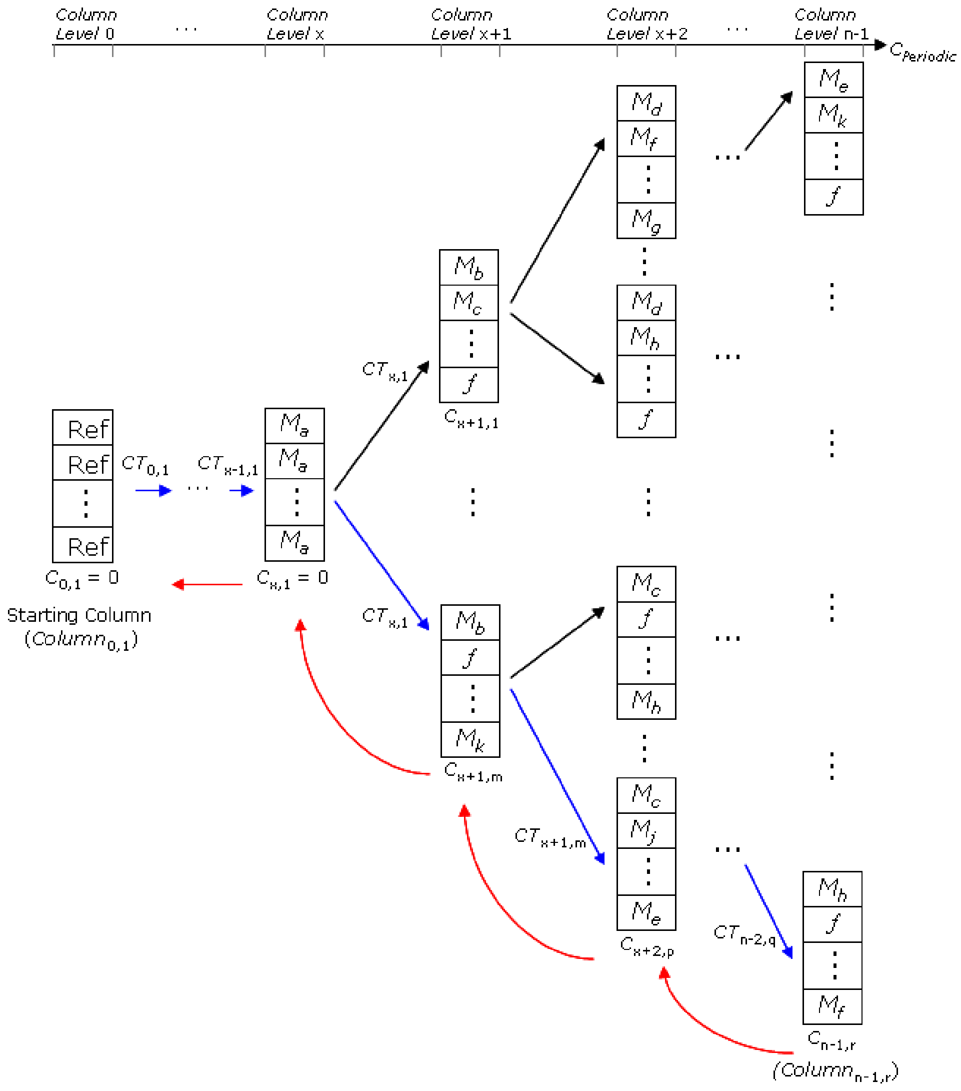


Figure 4.2 Implementation of the method to obtain optimum SM design for an ideal message set with unequal messages lengths

As shown in Figure 4.2, each level of the tree consists of column elements with different combinations of message placement. As stated earlier, having *n* levels in the tree means that *n* columns are allocated in the SM for periodic message scheduling. Cost values, $c_{i,j}$ denote in-window loss (in $\mu\sec$) in the corresponding column element that with the index representation of j^{th} column element in the i^{th} tree level, and it can be expressed as

$$c_{i,j} = \sum_{m \in CE_{i,j}} (l_{\max i,j} - l_m) \cdot k_m \cdot \tau_{bit} \,, \tag{4.5}$$

where *m* denotes the messages in the set, $CE_{i,j}$ is defined as the set of messages that appear as placed in the corresponding column, and $l_{\max i,j}$ denotes the largest message size (in bits) in $Column_{i,j}$. Also, k_m is the number of appearances of periodic messages placed in $Column_{i,j}$ and simply it can be calculated by Equation (4.2) with $k_m = \dfrac{T}{pm_m}$.

As observed from Equation (4.5) $c_{i,j}$ relates the total difference between transmission times of the message with maximum size, which determines the column width ($cw_{i,j}$) and other messages in $Column_{i,j}$. So, Equation (4.5) can be equally rewritten by using Equation (3.3) that gives transmission time of a message.

$$c_{i,j} = \sum_{m \in CE_{i,j}} (tm_{\max i,j} - tm_m) \cdot k_m \,, \tag{4.6}$$

In Figure 4.2, first *x* levels of the tree have only one column element and $c_{i,j}$ values for them are shown as zero, since all windows in each are assigned to messages (i.e. M_a) that has the period of *B*, implying no alternative message arrangement combination for these columns.

Another notation, $CT_{i,j}$ stands for the total cost value that is the sum of cost values ($c_{i,j}$) of the column elements on the path from the starting column ($Column_{0,1}$) to corresponding column element ($Column_{i,j}$). If $P_{i,j}$ is defined as the set of parent columns on the path up to $Column_{i,j}$ including itself, then $CT_{i,j}$ can be expressed as

$$CT_{i,j} = \sum_{(Column_{a,b}) \in P_{i,j}} c_{a,b} \quad , \tag{4.7}$$

Since the index variable *i* shows the level number on the path $P_{i,j}$, there exist *i* column elements, resulting in *i* cost values ($c_{i,j}$). An example path is shown in Figure 4.2 with blue arrows from initial column (column with the reference message) to $Column_{n-1,r}$.

By this approach, not only messages are considered to be placed in windows, but also some windows in columns are remained as free window (but limited not to exceed $C_{Periodic}$) for the sake of obtaining all possible SM constructions. Thus, this workflow of procedure makes the algorithm have the implementation complexity of $\Theta(L^G)$, where *L* denotes the number of lines of the SM and *G* is the number of messages in the message set M_P.

For the illustration in Figure 4.2, let's say that the last tree column element with the cost value, $c_{n,r}$ satisfies the minimum bandwidth loss with having the final total cost value $CT_{n-1,r} = CT_{n-2,q} + c_{n-1,r}$. And by applying back tracing as shown with red arrows, other previous parent columns can be obtained, resulting in SM construction with minimum in-window loss.

Figure 4.3 illustrates an example to make the workflow of the procedure more clear with using an example message set that consists of 4 periodic messages with the following periods and sizes (in bits): $M_1 : pm_1, 95$; $M_2 : 2pm_1, 65$; $M_3 : 2pm_1, 135$ and $M_1 : 2^2 pm_1, 65$. In addition to message properties, allocated column width is assumed as $C_{Periodic} = 1000 \ \mu\text{sec}$. Since it is an ideal message set, B and L values can be obtained as $B = pm_1$ and $L = 4$, which makes $T = 4pm_1$. The notation "*f*" stands for free windows in the columns.

As shown in Figure 4.3, at first the column that contains reference message is generated without any parent column *id.* and storing the following specific properties and information: $Column_{0,1}$, unplaced message list: $\{M_1, M_2, M_3, M_4\}$, $CT = 0$ and $C_{used} = 190 \ \mu\text{sec}$. Then the construction of the tree structure begins with taking the reference column ($Column_{0,1}$) under consideration. Since the first message in its unused message list is M_1, it is chosen to be placed in generated child columns. With having $k_1 = \frac{T}{pm_1} = 4$, M_1 remains no windows in the new column for other message placements, which result in only one message configuration as providing with $Column_{1,1}$. This column is generated with the following specific properties: $Column_{1,1}$, parent column: $Column_{0,1}$, unused message list: $\{M_2, M_3, M_4\}$, $CT = 0$ and $C_{used} = 301$. After that $Column_{1,1}$ is taken under consideration as being the next column with its specified properties. Since the first message in its unplaced message list is M_2, it is chosen and three child columns with different message configuration are generated that, all containing M_2 as being the message chosen from the list at the beginning. For example, the newly generated tree element,

$Column_{2,3}$ contains the following properties: $Column_{2,3}$, parent column: $Column_{1,1}$, unused message list: $\{M_3\}$, $CT = 0$ and $C_{used} = 463$. The same procedure is applied for all columns elements of the tree until there remains no unplaced message for each of them. Yet, during this process as mentioned before, the columns that exceeds $C_{Periodic}$ in the duration of its C_{used} value are not generated. For instance, $Column_{4,1}$ is not generated since its C_{used} value is 1038 μsec that is larger than the defined value of $C_{Periodic}$. After the construction of tree with all possible columns that form system matrices, leaf column elements are obtained and added to leaf column array. Although $Column_{3,1}$ seems as a leaf element, it is not added to the array since $Column_{3,1}$ still has M_4 unplaced. Because of that, this leaf element is ignored for the investigation of the optimum SM. By this way last three columns ($Column_{3,2}$, $Column_{3,3}$ and $Column_{3,4}$) are added to leaf column array. After the search for the optimum SM that provides minimum CT value, $Column_{3,4}$ becomes the winner with having zero total cost value (CT), which equally means zero in-window loss. Figure 4.3 also shows the resulting SM that is obtained with combining the parent columns on the path with blue color.

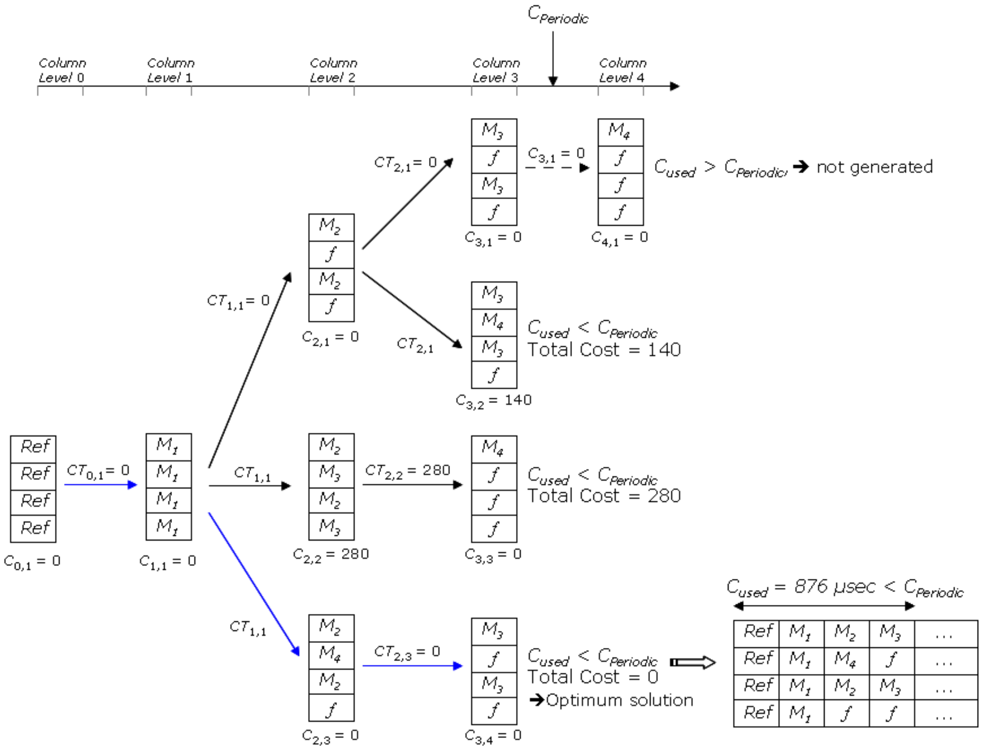
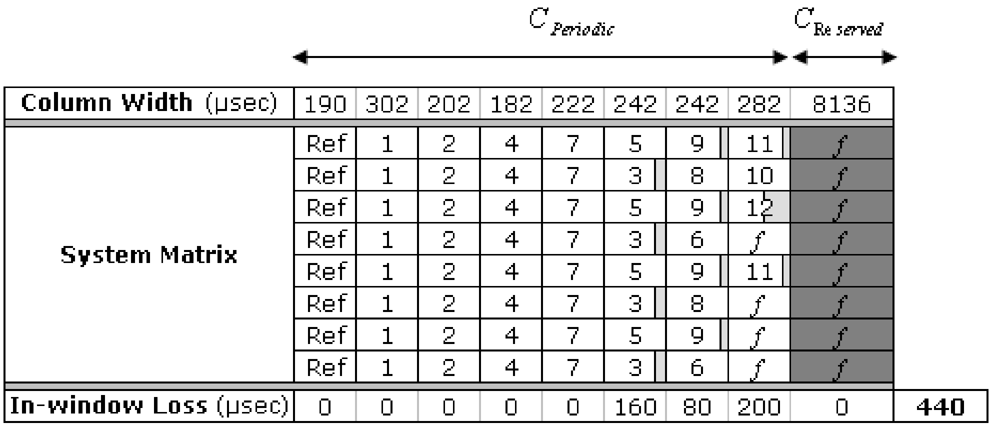


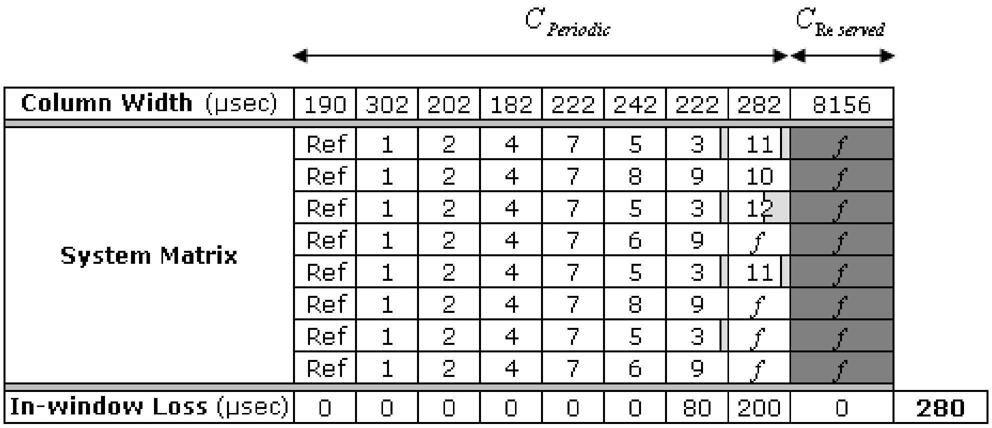
Figure 4.3 Optimum SM construction to provide minimum in-window loss for an example periodic message set

In the following example, two possible solutions are applied to PSA benchmark message set given in Table 3.1 by using the TTCAN scheduler tool. Firstly, the SM is constructed with using the first approach by ordering messages in the SM with respect to their periods as if message sizes are equal (Figure 4.4). The second SM is constructed by the proposed method to obtain minimum in-window loss, which is the optimum solution considering bandwidth loss (Figure 4.5). Grey parts in the windows of the system matrices show the duration during which there is no transmission; that is, they represent in-window loss in μsec caused by unequal message lengths.


$C_{Periodic}$
$C_{Reserved}$

Column Width (µsec)	190	302	202	182	222	242	242	282	8136	
System Matrix	Ref	1	2	4	7	5	9	11	*f*	
	Ref	1	2	4	7	3	8	10	*f*	
	Ref	1	2	4	7	5	9	12	*f*	
	Ref	1	2	4	7	3	6	*f*	*f*	
	Ref	1	2	4	7	5	9	11	*f*	
	Ref	1	2	4	7	3	8	*f*	*f*	
	Ref	1	2	4	7	5	9	*f*	*f*	
	Ref	1	2	4	7	3	6	*f*	*f*	
In-window Loss (µsec)	0	0	0	0	0	160	80	200	0	**440**

Figure 4.4 The SM designed by using the first approach for PSA benchmark message set

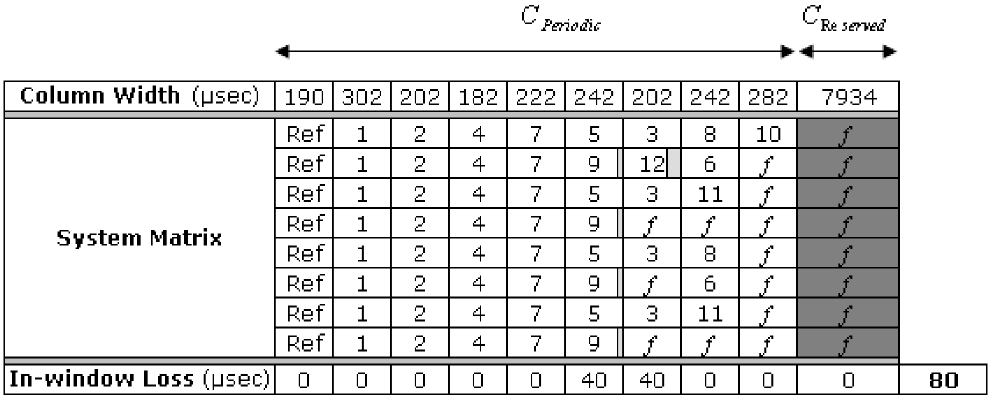

$C_{Periodic}$
$C_{\mathrm{Re}served}$

Column Width (μsec)	190	302	202	182	222	242	222	282	8156	
System Matrix	Ref	1	2	4	7	5	3	11	f	
	Ref	1	2	4	7	8	9	10	f	
	Ref	1	2	4	7	5	3	12	f	
	Ref	1	2	4	7	6	9	f	f	
	Ref	1	2	4	7	5	3	11	f	
	Ref	1	2	4	7	8	9	f	f	
	Ref	1	2	4	7	5	3	f	f	
	Ref	1	2	4	7	6	9	f	f	
In-window Loss (μsec)	0	0	0	0	0	0	80	200	0	**280**

Figure 4.5 The SM designed by using the second approach for PSA benchmark message set

In this example, bus bit rate is taken as 500 kbps that makes $\tau_{bit} = 2\,\mu\text{sec}$. The numbers in the SM denote message "*id*s" and "*f*" is used for free windows. Also, it is assumed that the last column with the width of $C_{\mathrm{Re}served}$ is left as free window for arbitration windows and future use as network extension or evolution so that, total column duration allocated for periodic message scheduling is designated as $C_{Periodic}$.

The first SM shown in Figure 4.4 outputs the network utilization and matrix load as follows, $NU = 25.77\%$ and $ML = 17.23\%$ by causing in-window loss (Bw_{loss}^{iw}) of $440\,\mu\text{sec}$ per matrix cycle. The SM, constructed by the second approach and given by Figure 4.5, has the same configuration proposed in [5] as an optimum solution. It provides the optimum result in terms of bandwidth loss by making it zero in the fifth column. The decrease in Bw_{loss}^{iw} makes *NU* increase and *ML* decrease by the Equations (3.17) and (3.18). The results are obtained as $NU = 26.07\%$ and $ML = 17.03\%$, which are better than *NU* (25.99%) and *ML* (17.08%) values for the SM configuration given in [43] and [44]. In addition, both designed matrices result in zero jitter and minimum number of triggers (total 13 *Tx_triggers*; 1 for reference trigger and 12 for other periodic messages) owing to ideal message set.


$C_{Periodic}$
$C_{\mathrm{Re}served}$

Column Width (μsec)	190	302	202	182	222	242	202	242	282	7934	
System Matrix	Ref	1	2	4	7	5	3	8	10	f	
	Ref	1	2	4	7	9	12	6	f	f	
	Ref	1	2	4	7	5	3	11	f	f	
	Ref	1	2	4	7	9	f	f	f	f	
	Ref	1	2	4	7	5	3	8	f	f	
	Ref	1	2	4	7	9	f	6	f	f	
	Ref	1	2	4	7	5	3	11	f	f	
	Ref	1	2	4	7	9	f	f	f	f	
In-window Loss (μsec)	0	0	0	0	0	40	40	0	0	0	**80**

Figure 4.6 The SM designed by using the second approach with increased duration allocated to periodic message scheduling

The total column width allocated to periodic message scheduling ($C_{Periodic}$) plays an important role in designing the SM. If the previous assumption about the width of reserved part ($C_{\mathrm{Re}served}$) is changed with decreasing it by $202\mu\sec$, an additional column could be introduced for periodic message scheduling. TTCAN scheduler gives the SM as shown in Figure 4.6, by which better performance is obtained as $NU = 26.38\%$ and $ML = 16.83\%$ with less amount of Bw_{loss}^{iw}. Thus, it is possible to obtain system matrices providing better real-time performance depending on the allocated SM duration.

4.4.2 SM Design for Non-Ideal Message Set

In this section, the discussion of SM design is extended for other periodic message sets that do not have the ideal message set property, which is message periods' being written in the form of $2^j pm_1$. In [5], some formal methods such as period reduction and matrix cycle reduction are proposed for scheduling this type of message sets to satisfy the hardware and protocol constraints. In this section, these methods in addition to basic cycle count reduction are analyzed and evaluated in the scope of real-time performance. The analysis is performed and results are obtained by the TTCAN scheduler tool, which help an appropriate SM design including system requirements. This approach proposes to construct the SM based on system requirement defined with priorities in terms of performance metrics.

As discussed in [5] with an example of SAE benchmark message set, this type of message set can also be scheduled with zero jitter by defining *T* as the least common multiple (*lcm*) of message periods with choosing the number of lines (*L*) according to TTCAN constraints and calculating *B* as T/L (Equation (3.6)). By this method zero jitter is obtained but with forcing the use of 55 *Tx_triggers* for only one node. The number of triggers does not obey the previously stated constraint expressed with Equation (3.12).

Also, in [5] some methods are proposed alternative to the previous approach, which are message period reduction, matrix cycle reduction and frame packing.

Reduced message periods: For a non-ideal periodic message set $M_P = \{M_1, M_2, ..., M_G\}$, message periods may be defined as $\{pm_1, pm_2, ..., pm_G\}$ and *N* denotes the number of nodes in the embedded network and for now message sizes are assumed to be equal. To satisfy the requirement of minimum number of triggers, at first the message period reduction method can be used. Reduced message periods are expressed as $\{mp_1, mp_2, ..., mp_G\}$, according to which the message set M_P will be scheduled.

Firstly, the basic cycle duration is taken as $B = pm_1$, where pm_1 is the smallest period in M_P. Then message periods in M_P can be reduced with modifying in the form of $2^j B$ to perform the SM design as if M_P is an ideal message set. So the reduced periods are expressed as $\{x_1 B, x_2 B, ..., x_G B\}$ where $X = 2^k = \{x_1, x_2, ..., x_G\}$, $k \in N$. At first, reduced periods of messages are defined as the closest smaller periods in the form of $2^j B$.

For this case the trigger count for M_1 can be expressed as, $T_{M_1} = 1 + (N-1) = N$, where "1" is for *Tx_trigger* and "$(N-1)$" for *Rx_trigger*. Since trigger count is same for other messages (including the reference message), total number of triggers of the networked system can be expressed as

$$T_{Total} = N \times (G+1), \tag{4.8}$$

where the expression $(G+1)$ stands for the number of messages including the reference messages exchanged in the network. Thus, Equation (4.8) implies $(G+1)$ triggers for each node so that the constraint should be satisfied as $(G+1) \leq 32$. Moreover, *L* can be calculated as $L = \frac{x_G B}{x_1 B} = x_G = 2^{k_G} = 2^q$ where $q \leq 6$ because of the constraint expressed in Equation (3.10).

Bandwidth loss in the SM can also be analyzed with substituting the derived equations. By using the expression of *L* and Equation (3.16), bandwidth loss caused by the reference message can be expressed as

$$Bw_{loss}^{ref} = x_G \cdot lm_{\mathrm{Re}f} \cdot \tau_{bit}, \tag{4.9}$$

To derive the expression for Bw_{loss}^{uw} (bandwidth loss caused by unused windows in the SM) and jitter, firstly message transmission instances should be defined. The number of expected transmission instances (time instant at which the message will be ready to be sent) during the duration of $lcm(pm_y, x_y B)$ for a generic message (M_y) in M_P can be expressed as

$$n_{i,y} = \frac{lcm(pm_y, x_y B)}{pm_y}, \tag{4.10}$$

where the term $lcm(pm_y, x_y B)$ relates the least common multiple of the actual and reduced message periods. Also, the number of actual transmission instants (time instant at which transmission of the message will begin based on the SM) of M_y during the duration of $lcm(pm_y, x_y B)$ can be given as

$$n_{w,y} = \frac{lcm(pm_y, x_y B)}{x_y B}, \tag{4.11}$$

Number of actual transmission instances of a message also defines the number of exclusive windows assigned to corresponding message in the SM. Since exclusive windows are assigned to messages with respect to their reduced periods, there exists no message transmission in some of the windows called as unused windows that result in bandwidth loss denoted as Bw_{loss}^{uw}. By this way, the difference between the terms $n_{i,y}$ and $n_{w,y}$ gives the number of unused windows for M_y so, Bw_{loss}^{uw} for the duration of total matrix duration (*T*) can be expressed as

$$Bw_{loss}^{uw} = \frac{(n_{w,y} - n_{i,y}) \cdot (lm_y + 16) \cdot \tau_{bit}}{lcm(pm_y, x_y B)} \cdot T , \qquad (4.12)$$

Finally, since equal message sizes are assumed, $Bw_{loss}^{iw} = 0$.

Expected transmission instances of a generic message (M_y) in the message set M_P can be shown as a set of W

$$W = \{w_1, w_2, ..., lcm(pm_y, x_y B)\} = \{0, pm_y, 2pm_y, ..., lcm(pm_y, x_y B)\}, \qquad (4.13)$$

and actual transmission instances (time instant at which transmission of the message will begin based on the SM) can be shown as

$$WT = \{wt_1, wt_2, ..., lcm(pm_y, x_y B)\} = \{0, x_y B, 2x_y B, ..., lcm(pm_y, x_y B)\}, \qquad (4.14)$$

Yet, *WT* also includes time instances for unused windows (UW), during which there occurs no message transmission because of the reduced message periods. In other words UW is the set of time instances for unused windows during the duration of $lcm(pm_y, x_y B)$. The set of actual message transmission instances excluding the instances of unused windows (WT_a) consists of the elements of $wt_{a,i} \in WT$ with satisfying $wt_{a,i} \geq w_i$, where $wt_{a,i}$ is the smallest instance value bigger than each expected transmission instance starting from w_1 during the duration of $lcm(pm_y, x_y B)$. So that the set of time instances for unused windows, UW can be expressed as $UW = WT - WT_a$. Since the set of unused window instance is the subset of WT ($UW \subset WT$), the actual message transmission instances (transmission begin instances excluding the instances of unused windows) can be expressed as

$$WT_a = WT - UW = \{wt_{a,1}, wt_{a,2}, ..., lcm(pm_y, x_y B)\}, \tag{4.15}$$

So that with Equation (3.7), message transmission delay (response delay) for M_y can be expressed as

$$D_y = \sum_{wt_{a,i} \le lcm(pm_y, x_y B)} \left| wt_{a,i} - w_i \right|, \tag{4.16}$$

Thus, jitter of each message in the set M_P can be calculated easily by using Equation (3.8). The sum gives the total jitter of the system (Equation (3.9)).

Increased cycle duration: By this operation it is aimed to reduce the bandwidth loss by decreasing *L* and so Bw_{loss}^{ref}. Now *B* is increased to analyze the effects on the performance metrics. New *B* value is chosen as, $B_{new} = x_{new} \cdot B$ where $x_{new} \in \{x_i, x_{i+1} ...., x_G\}$ and $x_i > x_1$. So number of *Tx_triggers* and *Rx_triggers* for messages with reduced periods of $mp_i < B_{new}$ can be calculated as, $Tx_{M_i} = \frac{x_{new} \cdot B}{mp_i} = \frac{x_{new} \cdot B}{x_i \cdot B} = \frac{x_{new}}{x_i} > 1$ and $Rx_{M_i} = \frac{x_{new}}{x_i} \cdot (N-1)$. Thus, the sum gives the total number of triggers which can be expressed as $T_{M_i} = N \cdot \frac{x_{new}}{x_i}$. By more formal representation total number of triggers for a generic message M_y is

$$T_{M_y} = \begin{cases} \frac{x_{new}}{x_y} \cdot N & mp_y < B_{new} \\ N & else \end{cases}, \tag{4.17}$$

As a result, increasing *B* also increases the trigger count for the nodes in the network. If bandwidth loss is taken under consideration, higher value of *B* causes a decrease in *L* to satisfy the same *T* as, $L = \frac{x_G \cdot B}{x_{new} \cdot B} = \frac{x_G}{x_{new}}$. Thus, bandwidth loss due to the reference message per matrix cycle (Bw_{loss}^{ref}) decreases compared to Equation (4.9) and it can be expressed as

$$Bw_{loss}^{ref} = \frac{x_G}{x_{new}} \cdot lm_{\mathrm{Re}\,f} \cdot \tau_{bit}, \qquad (4.18)$$

with $x_{new} > x_1$. Since there is no change on reduced message periods and *T*, Bw_{loss}^{uw} stays constant as expressed with Equation (4.12). Thus from Equation (3.15), total bandwidth loss per matrix cycle (Bw_{Loss}) decreases with resulting in higher *NU* and lower *ML* as expressed in Equations (3.17) and (3.18).

Jitter Reduction: Up to now performance metrics such as number of triggers, network utilization and matrix load are considered. As expressed with Equation (4.16) there exist response delays and so jitter for messages due to the modified (reduced) message periods. To decrease the jitter is possible by further decreasing reduced message periods as applied for SAE message set in [5] to the messages with the period of 100 msec (decreasing it to 20 msec instead of 40 msec). M_y is again considered as a generic message in M_P with the initial *B* value and if the reduced period of M_y is further reduced as , $mp_y^{'} = x_y^{'} B$ where $x_y^{'} \in \{x_i, x_{i+1} \ldots ., x_G\}$ with $x_y^{'} < x_y$ for $mp_y^{'} \geq B$ not to disturb idealized message set. If mp_y is to be reduced lower than *B*, the value of $mp_y^{'}$ may be any positive integer division of *B*. Considering the trigger count, if $mp_y^{'} < B$, number of triggers increases for M_y, $T_{M_y} = \frac{B}{mp_y^{'}} \cdot N$ compared to initial case given by Equation (4.8). As stated previously $mp_y = x_y \cdot B$ where $x_y = 2^{k_y}$ and $k_y \in N$ for the message M_y but for this case it is possible to reduce the message period lower than *B* so further reduced period of M_y can be expressed as

$$mp_y^{'} = \begin{cases} x_y^{'}B \;\; where \;\; x_y^{'} \in \{x_i, x_{i+1} ...., x_G\} \quad for \quad mp_y^{'} \geq B \\ \\ \dfrac{B}{x_y^{'}} \;\; where \;\; x_y^{'} \in N^{+} \qquad\qquad for \quad mp_y^{'} < B \end{cases}, \qquad (4.19)$$

By this way $n_{w,y}$ value changes as $n_{w,y}^{'} = \dfrac{lcm(pm_y, x_y B)}{x_y^{'} B}$. Because of the inequality of $x_y^{'} < x_y$, $n_{w,y}$ increases as $n_{w,y}^{'} > n_{w,y}$ that also implies the increase in the number of unused exclusive windows ($n_{w,y}^{'} - n_{i,y}$) assigned to M_y.Thus, Bw_{loss}^{uw} increases due to the first term in the numerator ($n_{w,y} - n_{i,y}$) in Equation (4.12). On the other hand, further message period reduction causes closer and denser assigned exclusive windows resulting in closer message transmission instances to each other for M_y, given in Equation (4.15), as $wt_{a,i}^{'} < wt_{a,i}$, which reveals lower value of D_y from Equation (4.16) and so lower message jitter.

Analysis of an example message set: The above performed theoretical analyses show that different scheduling methods result in different performance metrics. At this point, system performance requirements in addition to message properties and protocol constraints have an important role to decide the methods for SM design. It is possible to describe these requirements with defining its priorities in terms of performance metrics. The proposed approach is shown over a self constructed non-ideal message set example by performing analysis and evaluations with the help of TTCAN scheduler tool to designate the methods and roadmap for an appropriate SM design.

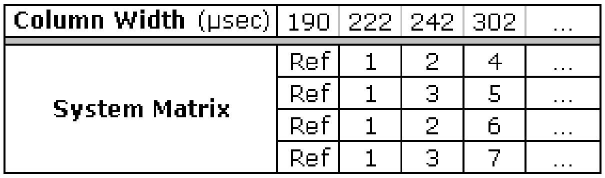
Table 4.2 gives a non-ideal message set example that consists of 7 periodic messages exchanged between 4 nodes, among which *Node* 1 is the master node.

Table 4.2 A self constructed non-ideal message set example

Message *id*.	**Length** (bits)	**Period** (μsec)	**Deadline** (μsec)	**Transmitter Node**
1	95	10000	10000	Node 1
2	65	20000	20000	Node 4
3	105	23000	23000	Node 4
4	105	40000	40000	Node 1
5	65	40000	40000	Node 2
6	115	45000	45000	Node 2
7	135	70000	70000	Node 3

As mentioned previously, non-ideal message sets are treated as if they are ideal to satisfy protocol constraints. The minimum period pm_1 can be defined as $10000\mu\sec$ that is the period of M_1. At first, other message periods that are not in the form of $2^j pm_1$ are modified with reducing to the closest value in the specified form. By this way, it is aimed to satisfy protocol constraints (number of lines and duration of basic cycle) with providing the minimum number of triggers.

Table 4.3 The SM to provide minimum number of triggers with the related performance results for the non-ideal message given in Table 4.2



Column Width (μsec)	190	222	242	302	...
System Matrix	Ref	1	2	4	...
	Ref	1	3	5	...
	Ref	1	2	6	...
	Ref	1	3	7	...

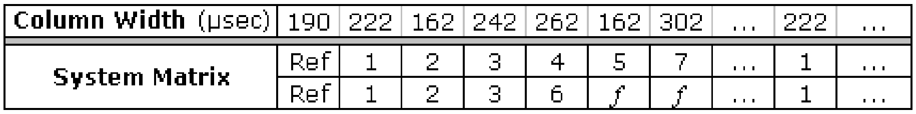
Message *id.*	pm_m (μsec)	mp_m (μsec)	T_M	J_m	$Bw_{Loss,m}$ (μsec)
Ref	10000	10000	4	0	760
1	10000	10000	4	0	0
2	20000	20000	4	0	160
3	23000	20000	4	41.30	63
4	40000	40000	4	0	60
5	40000	40000	4	0	140
6	45000	40000	4	38.89	69
7	70000	40000	4	21.43	129
Total			**32**	**101.62**	**1381**
$Bw_{Loss,avg}$ (%)	**3.45**				
NU (%)	**17.82**				
ML (%)	**9.56**				

In Table 4.3, pm_m and mp_m denote the actual and reduced period of messages respectively. T_M relates the total number of triggers assigned to the message in all 4 nodes and J_m is used for the jitter. On the other hand, $Bw_{Loss,m}$ relates the total amount of bandwidth loss for each matrix cycle. $Bw_{Loss,m}$ for the reference message is used to relate the bandwidth loss caused by reference windows (Bw_{loss}^{ref}), which is given by Equation (3.16), whereas $Bw_{Loss,m}$ for other messages is the sum of bandwidth loss due to unused windows (Bw_{loss}^{uw}) and in-window loss

(Bw^{iw}_{loss}) for each matrix cycle. As observed from the results, minimum trigger count is achieved but with a considerable amount of jitter and bandwidth loss.

In the case of high priority of low bandwidth loss as a system requirement, SM configuration should be changed with decreasing the number of basic cycles. In the second analysis, *L* is halved, while *B* is doubled and following results are obtained as given in Table 4.4.

Table 4.4 The SM after doubling *B* to provide less bandwidth loss and related performance results



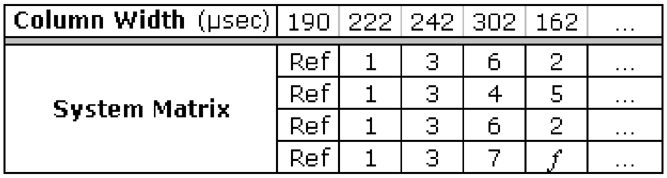
Column Width (µsec)	190	222	162	242	262	162	302	...	222	...
System Matrix	Ref	1	2	3	4	5	7	...	1	...
	Ref	1	2	3	6	*f*	*f*	...	1	...

Message *id.*	pm_m (µsec)	mp_m (µsec)	T_M	J_m	$Bw_{Loss,m}$ (µsec)
Ref	10000	10000	4	0	380
1	10000	10000	8	0	0
2	20000	20000	4	0	0
3	23000	20000	4	41.30	63
4	40000	40000	4	0	20
5	40000	40000	4	0	0
6	45000	40000	4	38.89	29
7	70000	40000	4	21.43	129
Total			**36**	**101.62**	**621**
$Bw_{Loss,avg}$ (%)	**1.55**				
NU (%)	**22.25**				
ML (%)	**7.66**				

Increasing *B* results in higher number of both Tx and Rx_triggers. Yet, smaller number of lines invokes less reference message transmission, which result in lower Bw_{loss}^{ref}. An interesting point is that smaller number of lines causes less amount of bandwidth loss for also other messages, since decreasing basic cycle count means less number of different messages placed in the same SM column, so less message size difference, resulting in less Bw_{loss}^{iw}. As observed from the performance results decreasing amount of total bandwidth loss (Bw_{Loss}) affects *NU* with a significant increase.

Different from the previous SM design, if jitter has high priority as a system performance requirement, SM configuration would be changed to provide less jitter. A solution may be proposed as further reducing the message periods that cause jitter, which results in higher number of exclusive windows reserved for messages in the SM. Table 4.5 gives the SM that is designed according to further reduced message periods and the related performance results.

Table 4.5 The SM after further period reduction to provide less jitter and related performance results



Column Width (μsec)	190	222	242	302	162	…
System Matrix	Ref	1	3	6	2	…
	Ref	1	3	4	5	…
	Ref	1	3	6	2	…
	Ref	1	3	7	*f*	…

Message *id.*	pm_m (μsec)	mp_m (μsec)	T_M	J_m	$Bw_{Loss,m}$ (μsec)
Ref	10000	10000	4	0	760
1	10000	10000	4	0	0
2	20000	20000	4	0	0
3	23000	10000	4	19.57	547
4	40000	40000	4	0	60
5	40000	40000	4	0	0
6	45000	20000	4	16.67	371
7	70000	40000	4	21.43	129
Total			**32**	**57.67**	**1867**
$Bw_{Loss,avg}$ (%)	**4.67**				
NU (%)	**15.81**				
ML (%)	**10.78**				

Reducing message periods means more exclusive windows assigned to the messages, which result in more unused windows and increasing amount of Bw_{loss}^{uw}. This reveals a significant decrease in *NU*. On the other hand, the designed SM reduces jitter at an important amount with making messages wait less time to be transmitted.

Analysis of SAE benchmark message set [5]: The performed analysis can also be applied to a real message set that is SAE periodic message set given by Table 4.1.

In the message set, $pm_1 = 5000\ \mu\sec$ and there are messages with periods of 100 and 1000 msec that are not in the form of $2^j\ pm_1$.

Firstly, B and L values are taken as $B = 5000\ \mu\sec$, $L = 4$, which implies that $T = 20000\ \mu\sec$. The message periods of 100 and 1000 msec can be reduced to closest period that can be written as $2^2\ pm_1$ with purpose of making the set ideal. Thus, Table 4.6 gives the resulting SM for reduced message periods and the related performance results.

Table 4.6 SM for SAE periodic message set after period reduction and related performance results

	B = 5000 *µsec*													
System Matrix	Ref	1	2	4	5	6	7	10	11	12	3	18	15	...
	Ref	1	2	4	5	6	7	10	11	13	8	19	20	...
	Ref	1	2	4	5	6	7	10	11	12	16	9	21	...
	Ref	1	2	4	5	6	7	10	11	13	17	14	22	...

Jitter	0
Total Tx_Trigger Count	50
$\boldsymbol{Bw_{Loss,avg}}$ (%)	12.45
NU (%)	7.00
ML (%)	42.68

Total Tx_trigger count value represents the sum of Tx and Rx_triggers for all messages including the reference message in the system. In spite of the reduced message periods, the SM results zero jitter because of that real message periods are integer multiples of reduced periods, which can be expressed as $lcm(pm_m, mp_m) = pm_m$. Also, minimum number of triggers is achieved by one Tx and Rx_trigger for each message. Since all the sizes of all messages are equal,

$Bw_{Loss}^{iw} = 0$. Except for reference messages, all bandwidth loss is caused by unused windows because of reduced message periods.

Secondly, to decrease the bandwidth loss, B value is doubled while keeping L constant, which result in $T = 40000\ \mu\sec$. Yet at this time, the message periods are reduced to $2^2 B$ which is equal to T value. Table 4.7 gives the performance results for the SM constructed according to defined parameters.

Table 4.7 SM for SAE periodic message set after doubling *B* and related performance results

	B = 10000 μsec															
System Matrix	Ref	1	2	4	5	6	7	10	11	12	13	12	3	18	15	...
	Ref	1	2	4	5	6	7	10	11	12	13	13	8	19	20	...
	Ref	1	2	4	5	6	7	10	11	12	13	12	16	9	21	...
	Ref	1	2	4	5	6	7	10	11	12	13	13	17	14	22	...

Jitter	60
Total Tx_Trigger Count	66
$Bw_{Loss,avg}$ (%)	5.69
NU (%)	8.31
ML (%)	35.92

Since the message period of 100 msec is not an integer multiple of the reduced period that is equal to T value, jitter increases. Also doubling B causes a significant increase in trigger count because of the messages with the period of $5000\ \mu\sec$. On the other hand, the SM results a significant decrease in bandwidth loss due to less proportion of time spent for the transmission of reference messages and less unused windows, because of which better *NU* and *ML* values are obtained.

Finally, to decrease the trigger count and jitter while not increasing bandwidth loss so much, B and L can be defined as $B = 5000\ \mu\text{sec}$ and $L = 8$ with further reducing the message periods of 100 msec to 20 msec again such as in the first configuration.

Table 4.8 SM for SAE periodic message set for less trigger count and jitter and related performance results

	B = 5000 μsec													
System Matrix	Ref	1	2	4	5	6	7	10	11	12	3	18	21	...
	Ref	1	2	4	5	6	7	10	11	13	8	19	22	...
	Ref	1	2	4	5	6	7	10	11	12	16	9	f	...
	Ref	1	2	4	5	6	7	10	11	13	17	14	f	...
	Ref	1	2	4	5	6	7	10	11	12	3	18	f	...
	Ref	1	2	4	5	6	7	10	11	13	8	19	f	...
	Ref	1	2	4	5	6	7	10	11	12	16	15	f	...
	Ref	1	2	4	5	6	7	10	11	13	17	20	f	...

Jitter	0
Total Tx_Trigger Count	50
$Bw_{Loss,avg}$ (%)	10.02
NU (%)	7.42
ML (%)	40.25

Again minimum trigger count and jitter values are obtained by this SM with the increased bandwidth loss but not as much as of the first SM configuration. Further reduction of message periods of 100 msec causes more unused windows in the SM that result in more bandwidth loss compared to second configuration.

In summary, different methods for SM design generate different performance results, which reveal the importance of the priorities of the system requirements on SM construction. Period reduction produces acceptable trigger count with simplifying the SM construction, yet it results in jitter and bandwidth loss by unused windows due to modified periods. Increased basic cycle duration decreases the number of lines of the SM and generates better *NU* due to less bandwidth loss caused by reference messages and in-window loss, yet it results in higher number of triggers. With applying jitter reduction, the message periods are further reduced, which results a significant decrease in jitter, yet it generates more bandwidth loss due to higher number of unused windows with providing less *NU*.

4.5 SM Design for Sporadic Messages

Up to now, event-triggered traffic that consists of sporadic messages has not been considered during the SM construction. Sporadic messages are non-periodic, event-triggered messages that are defined by their length (lm_m), minimum inter-arrival time (pm_m), and deadline (dm_m). Table 4.9 gives a SAE benchmark sporadic message set as an example, consisting of 31 messages and related message properties.

Table 4.9 SAE benchmark sporadic message set

Message *id*.	**Length** (bits)	**Min. Inter-arrival Time** (μsec)	**Deadline** (μsec)
1	65	50000	5000
2	65	20000	20000
3-31	65	50000	20000

Transmission of sporadic messages is performed in arbitration windows in the SM based on standard CAN arbitration. Thus, it is important to define number and arrangement of arbitration windows, which directly affects the system performance. By this way, SM design for sporadic messages could be divided into two main steps: 1) arbitration window count calculation, 2) arbitration window arrangement in SM.

Step 1: During SM design for sporadic messages, DM priority scheduling is assumed to be valid in arbitration windows that are equal in duration with each having one message transmission time. The method developed in [5] can be used for calculating the number of arbitration windows, which is also implemented in the TTCAN scheduler tool. The proposed algorithm in [5], uses message properties (period, deadline and size) and B as inputs to calculate the number of arbitration columns that consists of arbitration windows. Necessary number of columns is calculated by reserving arbitration windows for each sporadic message according to their deadlines, which guarantees the transmission of sporadic messages before their deadlines (schedulability). For instance, for SAE sporadic set given by Table 4.9, with $B = 5000\ \mu\sec$ the method produces the number of necessary arbitration columns as 8, but if B is taken as $B = 10000\ \mu\sec$, it produces 16 arbitration columns. By increasing B, the possibility of message arrivals during a BC also increases so that a higher number of arbitration columns are needed. The proposed approach in the thesis uses the algorithm for arbitration column count but assigns arbitration windows to all nodes as in CAN protocol instead of reserving arbitration windows for particular messages as proposed in [5]. In other words, all nodes can perform CAN arbitration during arbitration windows and the one with highest priority message wins the bus contention. By this way it is aimed to benefit from the advantages of event-triggered CAN protocol, which are flexibility with providing instant response in communication to significant events. Yet, assigning arbitration windows to all nodes causes a significant increase in trigger count due to the Tx_triggers. Also it should be noted that in this approach, guaranteeing schedulability of sporadic messages is not considered apart from arbitration column arrangements that it is assumed to have enough arbitration windows for sporadic messages. More specifically, under high load of traffics it may be possible for especially lower priority messages' missing their deadlines. By this way, the two approaches are compared considering only performance metric results as provided in Section 5.2.

Step 2: In the previous step the necessary number of arbitration columns is assumed to be available to schedule a sporadic message set. Yet, it is alone not enough to satisfy message set schedulability. *Step* 2 includes the operation of placement of previously calculated arbitration columns into the SM. It is important to satisfy schedulability from the arbitration column arrangement point of view. Arbitration window placement should be implemented considering two important rules as *rule* 1 and *rule* 2.

According to *rule* 1, the sum of the time interval between starting instances of any successive arbitration windows in the SM ($\Delta_{Arb(j,j+1)}$) and the transmission time of the message (tm_m) should not exceed the value of minimum deadline ($dm_{\min}$) in the sporadic message set. This inequality can be expressed as

$$\Delta_{Arb(j,j+1)} + tm_m \leq dm_{\min}, \tag{4.20}$$

Since DM priority scheduling is used for CAN arbitration, the message with minimum deadline is also the highest priority message in the message set; that is, this message is expected to be transmitted in the first arbitration window (here Arb_j) after its arrival (being ready to be transmitted). Yet, it is possible that the message would be blocked during the time of arbitration window by a lower priority message that has just before arrived. Because of this, in Equation (4.20) duration of the arbitration window Arb_j ($dur(Arb_j)$) is included in the left hand side of the inequality. This rule also works for other messages different from highest priority messages as follows; the time interval between the arrival of a message just before an arbitration window and transmission completion, not necessarily including successive arbitration windows, should not exceed the message deadline.

Another important point to be considered for arbitration windows arrangement is the number of critical sporadic messages that have relatively small deadlines such as, smaller than *B*. While defining distance between arbitration columns, it is also critical to provide necessary number of windows. This critical point is called as *rule 2* in the following parts for sake of simplicity. The following example makes this critical point clearer. For instance, a sporadic message set (M_S) includes two critical messages as M_1 and M_2 with the deadline of $dm_{\min}$, among which M_1 has the higher priority.

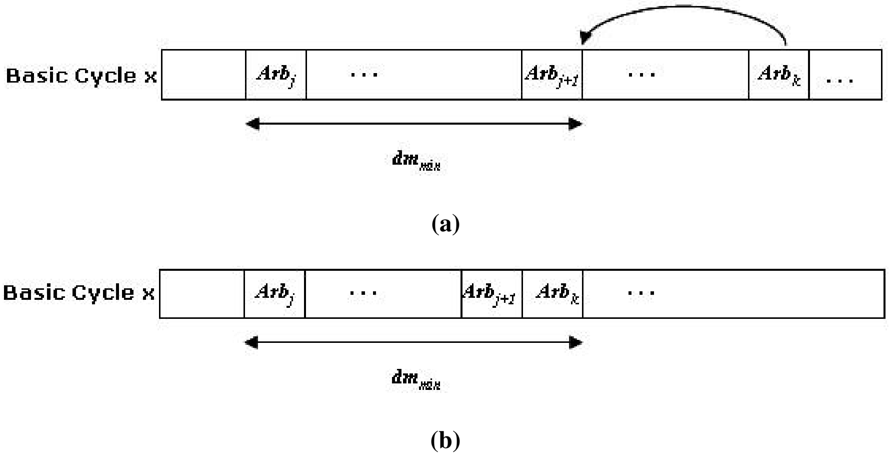


Figure 4.7 An example for arbitration window arrangement

As illustrated in Figure 4.7, Arb_j, Arb_{j+1} and Arb_k denote the successive arbitration columns, in which each arbitration window has the duration of only one sporadic message transmission time. In the first arrangement (a), it seems no problem according to the *rule* 1. Yet, as stated earlier we have two sporadic messages with the deadline of $dm_{\min}$ and they may possibly arrive at the same time just after the starting instant of the first arbitration window Arb_j. As a worst case scenario, again upon the blockage caused by a lower priority message during Arb_j, both critical messages have to compete in the next arbitration window Arb_{j+1}, which makes the message with the second priority (M_2) lose arbitration and miss its deadline. Yet, in the second arrangement, Arb_k is moved near Arb_j with the distance of $dm_{\min}$ from the beginning of Arb_j so that, deadline miss for M_2 under the mentioned worst case condition is eliminated.

Thus, arrangement of arbitration columns should be performed upon worst case response time analysis for all critical messages. The following algorithm helps to design the SM for sporadic messages with analyzing the configuration of arbitration columns in the SM with schedulability point of view. During the analysis, messages are assumed to be ready for transmission upon their arrivals without any queuing jitter. For the sporadic message set $M_S = \{M_1, M_2, ..., M_H\}$

```
set $i = 1$ and $j = 1$;
1) for each message $M_i \in M_S$ where $dm_i < B$        // loop 1
2)     for each arbitration column $Arb_j$                 // loop 2
3)         Search for an $Arb_k$ starting from $Arb_{j+1 \bmod (s+1)}$ where $k \neq j$ and
           $dur(Arb_k) \geq R_i$
           if (no $Arb_k$ found)
               end; // not schedulable due to the rule 2
           else if ( $Arb_k$ found)
               if $(\Delta_{Arb(j,k)} + R_i^{'} \leq dm_i)$
                   if (done for all arbitration columns)
                       if (done for all messages)
                           end; // schedulable for all messages
                       else
                           go to 1)
                   else
                       go to 2)
               else
                   end; // not schedulable due to rule 1
```

In M_S, messages are assumed to be ordered according to their deadline based on fixed priorities such that, M_1 has the highest priority for transmission on the bus. Also, in response time analysis given by the algorithm, adjacent arbitration columns are treated as one merged arbitration column. In 3), search for a suitable arbitration window Arb_k is performed in order, starting from Arb_j. The index parameter k take positive integer values of $\bmod(c+1)$, where c denotes the number of arbitration columns in the SM. Moreover, the term $n_{hp,i}$ denotes the number of message that have higher priority than M_i. During the search in the step 3) for the first arbitration column $Arb_{j+1 \bmod (s+1)}$, all higher priority messages are included in the response time analysis; yet in the following arbitration window in the case not obtaining an arbitration window with appropriate capacity, the higher priority messages with minimum inter-arrival times bigger than the duration of $\Delta_{Arb(j,k)} + dur(Arb_k)$ are not included since then $R_i^{'}$ becomes the total transmission times of number of higher priority messages that have new arrivals. Arbitration window durations are assumed to be relatively small compared to message minimum inter-arrival times (mit) so that, new arrivals of higher priority messages during an arbitration window are ignored. Also it should be noted that, under the cases of $dur(Arb_j) > R_i$, arbitration window duration is treated as if it has the length of R_i, which is necessary to create worst case conditions during analysis.

In summary, different from the proposed approach in [5], the arbitration windows are assigned for all sporadic messages that transmission of messages is performed by priority scheduling based on standard CAN arbitration. Yet, by the approach provided by the thesis the schedulability of sporadic messages, considering the number of arbitration windows, is not guaranteed that it is only assumed to have enough number of columns to satisfy the schedulability. This assumption is removed by the approach proposed by [5] with reserving arbitration windows for each sporadic message and so guaranteeing the schedulability in addition to resulting in less number of triggers. On the other hand, it is aimed better real-time communication performance in terms of delay and slack time metrics by providing instant response to significant events by standard CAN arbitration.

CHAPTER 5

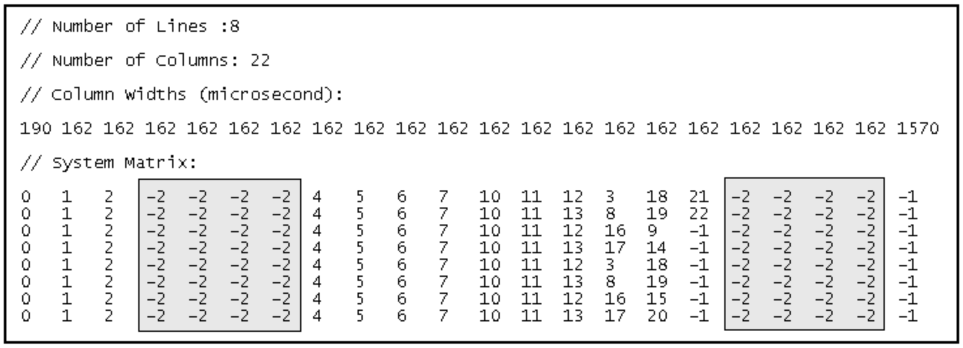
TEST RESULTS BY TTCAN SIMULATION

In Chapter 5, the TTCAN simulation tool is explained and some application examples are performed on the SAE benchmark message set with obtaining communication performance results by using this tool.

In previous chapters, sporadic messages were not considered during the SM analysis and performance evaluation. Only in Section 4.5, arbitration windows arrangement is explained to satisfy schedulability of sporadic messages. In this chapter, tests are performed including sporadic messages. Communication performance results are obtained in terms of previously defined performance metrics with the help of the TTCAN simulation tool that will be also described in the following section.

5.1 TTCAN Simulation Tool

The TTCAN simulation tool is developed in C++ programming language (visual studio 2003) with the main aim of evaluation of the SM performance in the scope of sporadic message scheduling. The tool uses both periodic and sporadic message properties (period, minimum inter-arrival time, deadline and size) and the SM structure as the inputs. During the simulation run, time and event-triggered traffics in the TTCAN network are simulated based on the previously constructed SM and it outputs the communication performance results, such as transmission delay, slack times and bus utilization. Figure 5.1 shows an example of SM input file for the TTCAN operation. The SM in the figure was previously designed (Table 4.8) for the SAE message set as given by Table 4.1 in Section 4.2 and now also arbitration columns are added to the SM, in which grey parts denote arbitration columns.



```
// Number of Lines :8
// Number of Columns: 22
// Column Widths (microsecond):
190 162 162 162 162 162 162 162 162 162 162 162 162 162 162 162 162 162 162 162 162 1570
// System Matrix:
0   1   2   -2  -2  -2  -2  4   5   6   7   10  11  12  3   18  21  -2  -2  -2  -2  -1
0   1   2   -2  -2  -2  -2  4   5   6   7   10  11  13  8   19  22  -2  -2  -2  -2  -1
0   1   2   -2  -2  -2  -2  4   5   6   7   10  11  12  16  9   -1  -2  -2  -2  -2  -1
0   1   2   -2  -2  -2  -2  4   5   6   7   10  11  13  17  14  -1  -2  -2  -2  -2  -1
0   1   2   -2  -2  -2  -2  4   5   6   7   10  11  12  3   18  -1  -2  -2  -2  -2  -1
0   1   2   -2  -2  -2  -2  4   5   6   7   10  11  13  8   19  -1  -2  -2  -2  -2  -1
0   1   2   -2  -2  -2  -2  4   5   6   7   10  11  12  16  15  -1  -2  -2  -2  -2  -1
0   1   2   -2  -2  -2  -2  4   5   6   7   10  11  13  17  20  -1  -2  -2  -2  -2  -1
```

Figure 5.1 An example of a SM input file for TTCAN simulation tool

The numbers in the SM denotes the message *ids*, for which the exclusive windows are reserved, whereas the numbers “0”, “-1” and “–2” stand for reference, free and arbitration windows respectively. For the SM in the figure bus bandwidth is taken as 500 kb/s that implies $\tau_{bit} = 2\ \mu\sec$. Since all messages (apart from the reference message) including the sporadic ones are equal in size with having 65 bits, the column widths are also equal in duration with having 162 $\mu\sec$, which is the transmission time of each message by Equation (3.3). During simulation run, all message traffic is performed according to the SM. Time-triggered messages are transmitted during their reserved exclusive windows, and defined event-triggered messages are transmitted during arbitration windows based on standard CAN arbitration.

As illustrated in Figure 5.1, arbitration columns are placed in the SM with the number of 8 as two distributed blocks. Since there is no critical sporadic message in the SAE message set with property of $dm_i < B$, deadline miss of messages due to arbitration column arrangement is prevented. In arbitration windows, DM priority scheduling policy is assumed to be applied and message transmission errors and faults in nodes are ignored during TTCAN operation.

Sporadic message arrivals is a random traffic and two different arrival traffics (arrival times and number of messages) are modeled for the sporadic messages of the system, based on uniform and poisson probability distribution. In the uniform traffic, sporadic message arrivals are determined during the time interval of respective minimum inter-arrival time of each message with the arrival probability of 0.2. Minimum inter-interval times of sporadic messages define the minimum time difference between successive arrivals. Similarly, poisson arrivals, as a non-uniform traffic, are determined based on poisson distribution with minimum inter-arrival times between successive arrivals. In the simulation, the arrived sporadic messages are assumed to be ready for transmission without any queuing jitter.

Upon the completion of the simulation, performance results such as number of messages that miss their deadlines, total delay and slack times, are provided for the evaluation of the SM with sporadic message scheduling point of view. Of course, it is also possible to obtain the results about the time-triggered traffic, yet performance results of the SM about periodic message scheduling have already been obtained by TTCAN scheduler tool.

5.2 Application Examples

In this section, the system matrices with different arbitration window configurations are tested and evaluated by the TTCAN simulation tool. SAE benchmark message set is used for these tests, so that the system matrices constructed for SAE periodic set given by Table 4.1, are taken under consideration.

Application 1: In this example uniform and poisson message traffics are compared by using the SM given in Figure 5.1. Table 5.1 gives the performance results of the SM under two different message traffics.

Table 5.1 Performance results of the SM given in Figure 5.1 under two different arrival traffics of SAE sporadic messages

	Uniform		**Poisson**	
No. Success	sporadic	periodic	sporadic	periodic
	1100	20661	1038	20661
No. Miss	0		0	
Tot. Delay (ms)	952		920	
Min. Slack (μsec)	2821		2895	
Tot. Slack (sec)	20.24		19.15	
BU (%)	35.81		35.71	

In Table 5.1, the terms "*No. Success*" and "*No. Miss*" denote the number of successful and unsuccessful transmissions due to deadline miss of messages during TTCAN operation. If the response time of a message is greater than the deadline, it is defined as unsuccessful transmission. The metrics total delay, minimum slack time and total slack time are only measured for the sporadic messages to evaluate the real-time performance and schedulability degree of the event-triggered traffic for the SM under consideration. The results given in Table

5.1 show that different from the uniform traffic, poisson arrivals result in more transmission delay in addition to less minimum and total slack time values although it imposes less number of message arrivals to the system. As expressed in Equation (2.3), more transmission delay means more queuing time spent before transmission. Also by Equation (3.19), smaller value of slack time implies that the time instant at which transmission of a message is complete is closer to its deadline. To sum up, by these performance results, it can be inferred that schedulability of sporadic messages with poisson arrivals is more difficult (that is, less schedulability degree) than ones with uniform arrivals. In the following examples poisson arrivals will be assumed for sporadic message traffic. However, it is not really critical if schedulability of the message set is guaranteed by the SM.

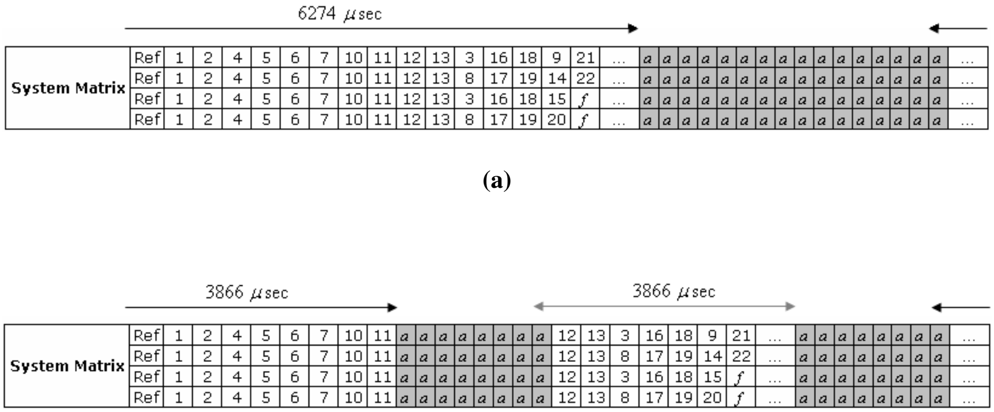
Application 2: In this example, the previously mentioned approaches proposed by [5] (I) and the thesis (II) in sporadic message scheduling are compared in terms of communication performance results. The test is performed based on the SM configuration in Figure 5.1 with using the poisson traffic.

Table 5.2 Performance results of the SM given in Figure 5.1 with using two approaches for sporadic message scheduling for the SAE sporadic message set

	(I)		**(II)**	
No. Success	sporadic	periodic	sporadic	periodic
	1038	20661	1038	20661
No. Miss	0		0	
Tot. Delay (ms)	3038		920	
Min. Slack (μsec)	144		2895	
Tot. Slack (sec)	17.03		19.15	
BU (%)	35.71		35.71	

As seen from the results, applying standard CAN arbitration to arbitration windows generates better real-time communication performance with less transmission delay and higher slack time values. On the other hand, the first approach guarantees no deadline miss for all traffic conditions by reserving arbitration windows for each of sporadic messages and it generates less number of triggers by providing total 31 Tx_triggers for all nodes compared to 40 for the second approach.

Application 3: In the third example the SM given in Figure 5.2 with two different arbitration column arrangements is tested. The terms "*a*" and "*f*" denote arbitration and free windows respectively. Since B value is taken as $B = 10000\ \mu\sec$, number of necessary arbitration columns to satisfy schedulability of SAE sporadic set has been previously calculated as 16 in Section 4.5. The B value of $10000\ \mu\sec$ makes the SAE sporadic set have one critical message that is the highest priority message in the set with the minimum deadline $dm_{\min}$ of $5000\ \mu\sec$ satisfying the property of $dm_{\min} < B$. Thus, arrangement of arbitration columns should be performed according to $dm_{\min}$ value following the rules provided in Section 4.5. As seen in the figure these arbitration columns are placed in the SM with two different configurations: just one block of 16 columns and two distributed blocks of 8 arbitration columns. Double way arrows in the figure show the difference between arbitration blocks, Δ_{Arb}. In (a) there is only one possibility for Δ_{Arb} as the time difference between the starting instances of the last column and the first column of the arbitration block, which is calculated as $6274\ \mu\sec$. Since the sum of this difference and the message transmission time is bigger than $dm_{\min}$ as $6427 + 162 > dm_{\min} = 5000\ \mu\sec$, the requirement provided by Equation (4.20) is not satisfied for the arbitration window arrangement in (a), which may result in the deadline miss cases of the critical message. On the other hand, in (b) the arbitration columns are placed with not exceeding $dm_{\min}$ value that guarantees the schedulability of the critical message.



System Matrix	Ref	1	2	4	5	6	7	10	11	12	13	3	16	18	9	21	...	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	...
	Ref	1	2	4	5	6	7	10	11	12	13	8	17	19	14	22	...	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	...
	Ref	1	2	4	5	6	7	10	11	12	13	3	16	18	15	*f*	...	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	...
	Ref	1	2	4	5	6	7	10	11	12	13	8	17	19	20	*f*	...	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	...

(a)

System Matrix	Ref	1	2	4	5	6	7	10	11	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	12	13	3	16	18	9	21	...	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	...
	Ref	1	2	4	5	6	7	10	11	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	12	13	8	17	19	14	22	...	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	...
	Ref	1	2	4	5	6	7	10	11	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	12	13	3	16	18	15	*f*	...	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	...
	Ref	1	2	4	5	6	7	10	11	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	12	13	8	17	19	20	*f*	...	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	...

(b)

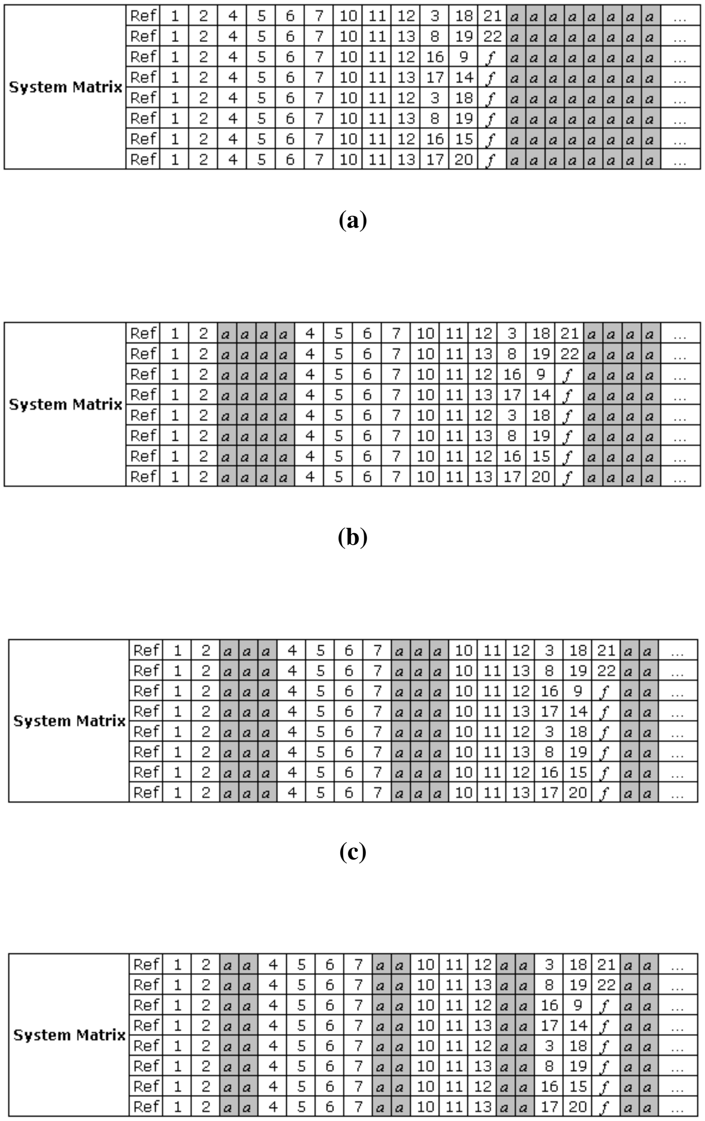
Figure 5.2 The SM with two different arbitration column arrangements

Since the SAE set have only one message with the property of $dm_{\min} < B$ arrangement of arbitration windows is implemented based on $dm_{\min}$ value. If there existed more than one critical message in the set, arrangement operation would be performed including other critical messages. The performance results given in Table 5.3 also support the above discussion that there exist 11 deadline miss for the critical message for the first SM since in (a) arrangement of arbitration columns does not obey *rule* 1 given in Section 4.5. Also, the SM in (b) generates better performance results considering other metrics with having smaller total delay and greater slack time values.

Table 5.3 Performance results for the system matrices with two different arrangements of arbitration columns as given by Figure 5.2

	SM (a)		**SM (b)**	
No. Success	sporadic	periodic	sporadic	periodic
	1027	19661	1038	19661
No. Miss	11		0	
Tot. Delay (ms)	3073.53		1680.49	
Min. Slack (μsec)	-63		1087	
Tot. Slack (sec)	16.94		18.39	
BU (%)	33.80		33.81	

Application 4: In this final example the SM given in Figure 5.1 is tested with different arrangements of arbitration columns. Figure 5.3 illustrates the SM with four different arbitration configurations. Having the B value of $5000\,\mu$sec imposes 8 arbitration columns for SAE sporadic set.



System Matrix	Ref	1	2	4	5	6	7	10	11	12	3	18	21	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	…
	Ref	1	2	4	5	6	7	10	11	13	8	19	22	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	…
	Ref	1	2	4	5	6	7	10	11	12	16	9	*f*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	…
	Ref	1	2	4	5	6	7	10	11	13	17	14	*f*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	…
	Ref	1	2	4	5	6	7	10	11	12	3	18	*f*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	…
	Ref	1	2	4	5	6	7	10	11	13	8	19	*f*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	…
	Ref	1	2	4	5	6	7	10	11	12	16	15	*f*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	…
	Ref	1	2	4	5	6	7	10	11	13	17	20	*f*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	*a*	…

(a)

System Matrix	Ref	1	2	*a*	*a*	*a*	*a*	4	5	6	7	10	11	12	3	18	21	*a*	*a*	*a*	*a*	…
	Ref	1	2	*a*	*a*	*a*	*a*	4	5	6	7	10	11	13	8	19	22	*a*	*a*	*a*	*a*	…
	Ref	1	2	*a*	*a*	*a*	*a*	4	5	6	7	10	11	12	16	9	*f*	*a*	*a*	*a*	*a*	…
	Ref	1	2	*a*	*a*	*a*	*a*	4	5	6	7	10	11	13	17	14	*f*	*a*	*a*	*a*	*a*	…
	Ref	1	2	*a*	*a*	*a*	*a*	4	5	6	7	10	11	12	3	18	*f*	*a*	*a*	*a*	*a*	…
	Ref	1	2	*a*	*a*	*a*	*a*	4	5	6	7	10	11	13	8	19	*f*	*a*	*a*	*a*	*a*	…
	Ref	1	2	*a*	*a*	*a*	*a*	4	5	6	7	10	11	12	16	15	*f*	*a*	*a*	*a*	*a*	…
	Ref	1	2	*a*	*a*	*a*	*a*	4	5	6	7	10	11	13	17	20	*f*	*a*	*a*	*a*	*a*	…

(b)

System Matrix	Ref	1	2	*a*	*a*	*a*	4	5	6	7	*a*	*a*	*a*	10	11	12	3	18	21	*a*	*a*	…
	Ref	1	2	*a*	*a*	*a*	4	5	6	7	*a*	*a*	*a*	10	11	13	8	19	22	*a*	*a*	…
	Ref	1	2	*a*	*a*	*a*	4	5	6	7	*a*	*a*	*a*	10	11	12	16	9	*f*	*a*	*a*	…
	Ref	1	2	*a*	*a*	*a*	4	5	6	7	*a*	*a*	*a*	10	11	13	17	14	*f*	*a*	*a*	…
	Ref	1	2	*a*	*a*	*a*	4	5	6	7	*a*	*a*	*a*	10	11	12	3	18	*f*	*a*	*a*	…
	Ref	1	2	*a*	*a*	*a*	4	5	6	7	*a*	*a*	*a*	10	11	13	8	19	*f*	*a*	*a*	…
	Ref	1	2	*a*	*a*	*a*	4	5	6	7	*a*	*a*	*a*	10	11	12	16	15	*f*	*a*	*a*	…
	Ref	1	2	*a*	*a*	*a*	4	5	6	7	*a*	*a*	*a*	10	11	13	17	20	*f*	*a*	*a*	…

(c)

System Matrix	Ref	1	2	*a*	*a*	4	5	6	7	*a*	*a*	10	11	12	*a*	*a*	3	18	21	*a*	*a*	…
	Ref	1	2	*a*	*a*	4	5	6	7	*a*	*a*	10	11	13	*a*	*a*	8	19	22	*a*	*a*	…
	Ref	1	2	*a*	*a*	4	5	6	7	*a*	*a*	10	11	12	*a*	*a*	16	9	*f*	*a*	*a*	…
	Ref	1	2	*a*	*a*	4	5	6	7	*a*	*a*	10	11	13	*a*	*a*	17	14	*f*	*a*	*a*	…
	Ref	1	2	*a*	*a*	4	5	6	7	*a*	*a*	10	11	12	*a*	*a*	3	18	*f*	*a*	*a*	…
	Ref	1	2	*a*	*a*	4	5	6	7	*a*	*a*	10	11	13	*a*	*a*	8	19	*f*	*a*	*a*	…
	Ref	1	2	*a*	*a*	4	5	6	7	*a*	*a*	10	11	12	*a*	*a*	16	15	*f*	*a*	*a*	…
	Ref	1	2	*a*	*a*	4	5	6	7	*a*	*a*	10	11	13	*a*	*a*	17	20	*f*	*a*	*a*	…

(d)

Figure 5.3 The SM with four different arbitration arrangements

As seen from the results given in Table 5.4, there is no deadline miss especially for the SM in (a), since there is no message in SAE set with the deadline smaller than B value, which is $5000\ \mu\text{sec}$.

Table 5.4 Performance results for the system matrices given by Figure 5.3

	SM (a)		**SM (b)**	
No. Success	sporadic	periodic	sporadic	periodic
	1038	20661	1038	20661
No. Miss	0		0	
Tot. Delay (ms)	1650.77		920	
Min. Slack (μsec)	1275		2895	
Tot. Slack (sec)	18.41		19.15	
BU (%)	35.71		35.71	
	SM (c)		**SM (d)**	
No. Success	sporadic	periodic	sporadic	periodic
	1038	20661	1038	20661
No. Miss	0		0	
Tot. Delay (ms)	820.96		778.47	
Min. Slack (μsec)	2895		2895	
Tot. Slack (sec)	19.25		19.29	
BU (%)	35.71		35.71	

Finally, the test results show that more common arbitration column distribution provides better real time performance for sporadic messages with having smaller transmission delays and greater slack times.

CHAPTER 6

CONCLUSION

6.1 Summary of the Thesis

Time-triggered nature of TTCAN brings important advantages compared to the standard CAN communication protocol. The advantages of dependability, robustness and predictability makes TTCAN protocol feasible for especially real-time embedded systems of safety critical applications such as, x-by-wire applications. Such systems obtain hard real-time messages exchanged between networked nodes, which possess strict time requirements. TTCAN communication protocol responds these requirements successfully with supporting both event and time-triggered traffic and combining the advantages of both types of communication paradigms.

In TTCAN, time windows (exclusive windows) are reserved for hard real-time messages to provide schedulability with dependability and predictability. Yet, this necessitates a schedule that manages the exchange of both event and time triggered traffic. In TTCAN networks, the SM, as a precomputed and fixed schedule arranges all message traffic. Because of that, real-time communication performance of a TTCAN network highly depends on the structure of the SM, which makes the design of the SM crucial.

As stated before, the main theme of the thesis is the construction of the TTCAN communication schedule considering protocol constraints, message properties and communication performance requirements of the system and it contains extended work on TTCAN SM design in [5]. In this thesis, firstly the assumptions, message properties and protocol constraints are explained to define bounds and workspace of SM construction. Periodic message sets are classified as ideal and non-ideal based on the property of ideal message periods (power of 2 multiples of the smallest message period) as provided in [5] for efficient schedules. Moreover, performance metrics are designated to form the basis for the performance criteria. In addition to adopted ones, a new performance metric called bandwidth loss per matrix cycle and its three different types for TTCAN network are defined: bandwidth loss due to reference message, in-window loss and unused windows, which are also interrelated with other used performance metrics, such as network utilization and matrix load. By defining this performance metric it is aimed to use it for analysis and design of the SM for periodic message scheduling, which provides a comprehensive and clear performance evaluation of system matrices.

Secondly, the scheduler and simulation tools have been developed in the scope of TTCAN communication scheduling analysis and design. The adopted and proposed methods on SM design for periodic message scheduling are implemented in the TTCAN scheduler tool. The simulation tool is mainly used to evaluate SM performance in the scope of sporadic message scheduling. In this thesis, SM design is discussed for both periodic and sporadic message scheduling. In periodic message scheduling, a method is developed for ideal periodic message sets with unequal message lengths to solve the optimization problem of filling the SM columns with the similar sized messages with using message properties (period, deadline and size) and allocated duration for periodic message scheduling in the SM. The proposed method that is also implemented in TTCAN scheduler tool generates an optimum solution with minimizing bandwidth loss due to unequal message lengths by evaluating SM structures with different configurations of message placements with respect to in-window loss and then selecting the one with minimum bandwidth loss.

Thirdly, for non-ideal periodic message sets, the methods such as message period reduction, increased basic cycle length and jitter reduction are analyzed with using the designated performance metrics, and their effects on communication performance results are shown theoretically. Also, system matrices constructed by applying these methods on example message sets are analyzed and evaluated to demonstrate their effects on the performance metrics with using TTCAN scheduler tool. According to results obtained by analyses, period reduction produces acceptable trigger count with simplifying the SM construction, yet it results in jitter and bandwidth loss by unused windows due to modified periods. Increased basic cycle duration decreases the number of lines of the SM and generates better network utilization due to less bandwidth loss caused by reference messages and in-window loss, yet it results in higher number of triggers. With applying jitter reduction, the message periods are further reduced, which results a significant decrease in jitter, yet it generates more bandwidth loss due to higher number of unused windows with providing less network utilization. Thus, these analyses reveal the fact that communication performance requirements of a real-time system play an important role on the SM design. The thesis provides an approach for SM design as combining three important concepts, which are message properties, protocol constraints and priorities of communication performance requirements of the system in terms of performance metrics.

Finally, scheduling of sporadic messages that is defined as a random event-triggered traffic is discussed. Different from the approach proposed in [5], which reserves arbitration windows for each sporadic message to guarantee the schedulability of the sporadic message set, in the thesis a scheduling approach is proposed with assigning arbitration windows to all sporadic messages by applying standard CAN arbitration to preserve an important attribute of event-triggered CAN, which is the flexibility with having instant response to significant events. Owing to this advantage, the proposed approach can be expected to have better real-time performance with less message transmission delays and higher slack time values, but in trade of higher number of triggers and not guaranteeing the schedulability. Moreover, the proposed approach includes arbitration column arrangements on the SM with defining two rules not to cause any sporadic message deadline miss due to arbitration column placements. These rules define the arbitration column placements and numbers distributed on the SM. In addition, considering sporadic message scheduling some analyses called application examples are provided on SAE benchmark message set with using TTCAN simulation tool. At first, two sporadic scheduling approached are compared. Although the first approach [5] guarantees no deadline miss and provides less number of triggers, the second approach results in better communication performance in terms of transmission delays and slack times values. Secondly, the next analysis relates the importance of arbitration column arrangement on the SM. It is shown that sporadic messages' deadline misses are prevented by simply applying the first rule that states the time interval between successive arbitration columns should not exceed the minimum deadline in the sporadic message set. And finally, SM configurations with different arbitration column arrangements are analyzed. Based on the results, it is concluded that more common arbitration column distribution provides better real time performance for sporadic messages with having smaller transmission delays and greater value of slack times.

6.2 Future Work

Considering previous studies and the work in the thesis, there are possible future works on TTCAN SM design. As stated before, in the thesis DM priority scheduling is assumed to be applied on arbitration windows. Approximated EDF priority scheduling algorithms can also be applied for possible better performance results on scheduling of sporadic messages.

Also, developing a tool that automatically designs TTCAN system matrices based on communication performance requirements of an embedded real-time system would be useful in providing easiness in SM design also for non-ideal message sets. Moreover, similar communication scheduling approaches and tools would be developed for other high speed time-triggered protocols such as, FlexRay [16] and byteflight [13][17].

REFERENCES

[1] J.W.S. Liu, *Real-Time Systems*, Prentice Hall, 1st edition, 2000.

[2] M. Barr, “Embedded Systems Glossary”, Netrino Technical Library. [Online]. Available: http://www.netrino.com/Embedded-Systems/Glossary-E, 14 July 2008.

[3] P. Pop, P. Eles, and Z. Peng, “Analysis and Optimization of Heterogeneous Real-Time Embedded Systems”, *IEE Proc. -Comput. Digit. Tech.*, Vol. 152, No. 2, 2005.

[4] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, “Trends in Automotive Communication Systems”, *Proceedings of the IEEE*, Vol. 93, No. 6, 1204-1224, 2005.

[5] K. Schmidt, E.G. Schmidt, “Systematic Message Schedule Construction for Time-Triggered CAN”, *IEEE Transactions on Vehicular Technology*, 2007.

[6] “Class C application requirement considerations”, Society for Automotive Engineers, Tech. Rep. J2056/1, 1993.

[7] Intel Corp.. Introduction to in-vehicle. [Online]. Available: http://support.intel.com/design/auto/autolxbk.htm, 20 June 2008.

[8] LIN specification package, revision 2.0. [Online]. Available: http://www.lin-subbus.org, 13 July 2008.

[9] “Class B data communication network interface”, Society for Automotive Engineers J1850 Standard, 1996.

[10] CAN in automation. [Online]. Available: http://www.can-cia.org, 20 June 2008.

[11] MOST specification rev. 3.0. [Online]. Available: http://www.mostnet.de, 13 July 2008.

[12] “D2B/SMARTwireX technology overview”. [Online]. Available: http://www.candc.co.uk, 5 July 2008

[13] G. Cena, A. Valenzano, S. Vitturi, “Advances in automotive digital communications”, *Computer Standards & Interfaces*, No. 27, pp. 665-678, 2005.

[14] F. Simonot-Lion, “In car embedded electronic architectures: How to ensure their safety”, *5th IFAC International Conference Fieldbus Systems and Their Applications (FeT 2003)*, 2003.

[15] Time-triggered protocol TTP/C, high level specification document, protocol version 1.1. [Online]. Available: http://www.tttech.com, 14 June 2008.

[16] FlexRay communication system, protocol specification, version 2.0. [Online]. Available: http://www.flexray.com, 14 June 2008.

[17] J. Berwanger, M. Peller and R. Griessbach. (1999) A new high-performance data bus system for safety-related applications. [Online]. Available: http://www.byteflight.com/specification, 20 June 2008.

[18] A. Albert, "Comparison of event-triggered and time-triggered concepts with regard to distributed control systems", *Embedded World*, 2004.

[19] P. Welander. (2008) Fieldbus: Growing globally. [Online]. Available: http://www.controleng.com/article/CA6539014.html, 22 July 2008.

[20] L. Almeida, P. Pedreiras, J. Alberto, and G. Fonseca, "The FTT-CAN protocol: why and how", *IEEE Transactions on Industrial Electronics*, Vol. 49, No. 6, pp. 1189-1201, 2002.

[21] H. Kopetz, "A comparison of CAN and TTP", *Annual Reviews in Control*, No. 24, pp. 177-188, 2000.

[22] T. Meyerowitz, C. Pinello, A. Sangiovanni-Vincentelli, "A tool for describing and evaluating hierarchical real-time bus scheduling policies", *DAC 2003*, 2003.

[23] H. Kopetz, "A solution to an automotive control system benchmark", *IEEE*, pp. 154-158, 1994.

[24] *Road Vehicles-Low speed serial data communication-Part 3: Vehicle area network (VAN)*, ISO 11 519-3, 1994.

[25] CAN in automation. [Online]. Available: http://ww.can-cia.org, 13 July 2008.

[26] G. Leen and D. Heffernan, “TTCAN: A new time-triggered controller area network”, *Microprocessors and Microsystems*, Vol. 26, pp. 77-94, 2002.

[27] N. Navet, “Controller area network”, *IEEE Potentials*, pp. 12-14, 1998.

[28] M. Farsi, K. Ratcliff and M. Barbosa, “An overview of controller area network”, *Networking Systems, Computing & Control Engineering Journal*, pp. 113-120, June 1999.

[29] D.I. Katcher, S.S. Sathaye and J.K. Strosnider, “Fixed priority scheduling with limited priority levels”, *IEEE Transactions on Computers*, Vol. 44, No. 9, pp. 1140-1144, September 1995.

[30] A. Burns, K. Tindell and A.J. Wellings, “Fixed priority scheduling with deadlines prior to completion”, *IEEE Computer Society Press*, pp. 138-142, June1994.

[31] K.W. Tindell, H. Hansson and A.J. Wellings, “Analysing real-time communications: Controller area network (CAN)”, *Proceedings of Real-Time Systems Symposium*, pp. 259-263, December 1994.

[32] K. Tindell, A. Burns, “Guaranteeing message latencies on controller area network (CAN)”, *Proceedings of the 1st International CAN Conference*, pp. 2-11, September 1994.

[33] K. Tindell, A. Burns and A.J. Wellings, “Calculating controller area network (CAN) message response times”, Control Engineering Practice, Vol. 3, pp. 1163-1169, August 1995.

[34] R.I. Davis, A. Burns, R.J. Brill and J.J. Lukkien, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised", *Real-Time Syst (2007)*, Vol. 35, pp. 239-272, 2007.

[35] K.M. Zuberi and K.G. Shin, "Scheduling messages on controller area network for real-time CIM applications", *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 2, pp. 310-314, April 1997.

[36] K.M. Zuberi and K.G. Shin, "Design and implementation of efficient message scheduling for controller area network", *IEEE Transactions on Computers*, Vol. 49, No. 2, pp. 182-188, February 2000.

[37] A. Meschi, M. Di Natale and M. Spuri, "Earliest deadline message scheduling with limited priority inversion“, *Proceedings of the 4th WPDRTS*, pp. 87-94, 1996.

[38] M. Di Natale, "Scheduling the CAN bus with earliest deadline techniques", *IEEE*, pp. 259-268, 2000.

[39] A. Albert, R. Strasser and A. Trachtler, "Migration from CAN to TTCAN for a distributed control system", *9th International CAN Conference (ICC 2003)*, pp. 5-16, 2003.

[40] T. Fuhrer, B. Muller, W. Dieterle, F. Hartwich, R. Hugel, M. Walther and R. Bosch GmbH, "Time triggered communication on CAN (Time Triggered CAN-TTCAN), Proceedings of 7th International CAN Conference, 2000.

[41] B. Muller, T. Fuhrer, F. Hartwich, R. Hugel, H. Weiler and R. Bosch GmbH, “Fault tolerant TTCAN networks”, *Proceedings of the 8th International CAN Conference (ICC)*, pp. 2-9, 2002.

[42] F. Hartwich, B. Muller, T. Fuhrer, R. Hugel and R. Bosch GmbH, “CAN network with time triggered communication”, *Proceedings of the 7th International CAN Conference*, 2000.

[43] J. Fonseca, F. Coutinho and J. Barreiros, “Scheduling for a TTCAN network with a stochastic optimization algorithm”, *International CAN in Automation Conference*, pp. 10-16, 2002.

[44] F. Coutinho, J. Barreiros and J. Fonseca, “Scheduling for a TTCAN network with a stochastic optimization algorithm”, *4th IFAC FET’2001*, 2001.

[45] A. Albert and R. Hugel, “Heuristic scheduling concepts for TTCAN networks”, *International CAN in Automation Conference (ICC)*, 2005.

[46] R. Johannson, “Time and event triggered communication scheduling for automotive applications”, Chalmers Lindholmen University College, Goteborg, Sweden, Tech. Rep. Technical Report 17, 2004.

[47] M. Naughton and D. Heffernan, “SMART-Plan: A new message scheduler for real-time control networks”, ISSC 2005, September 2005.

[48] N. Navet, Y.Q. Song and F. Simonot, “Worst-case deadline failure probability in real-time applications distributed over controller area network”, *Journal of Systems Architecture*, Vol. 46, pp 607-617, 2000.