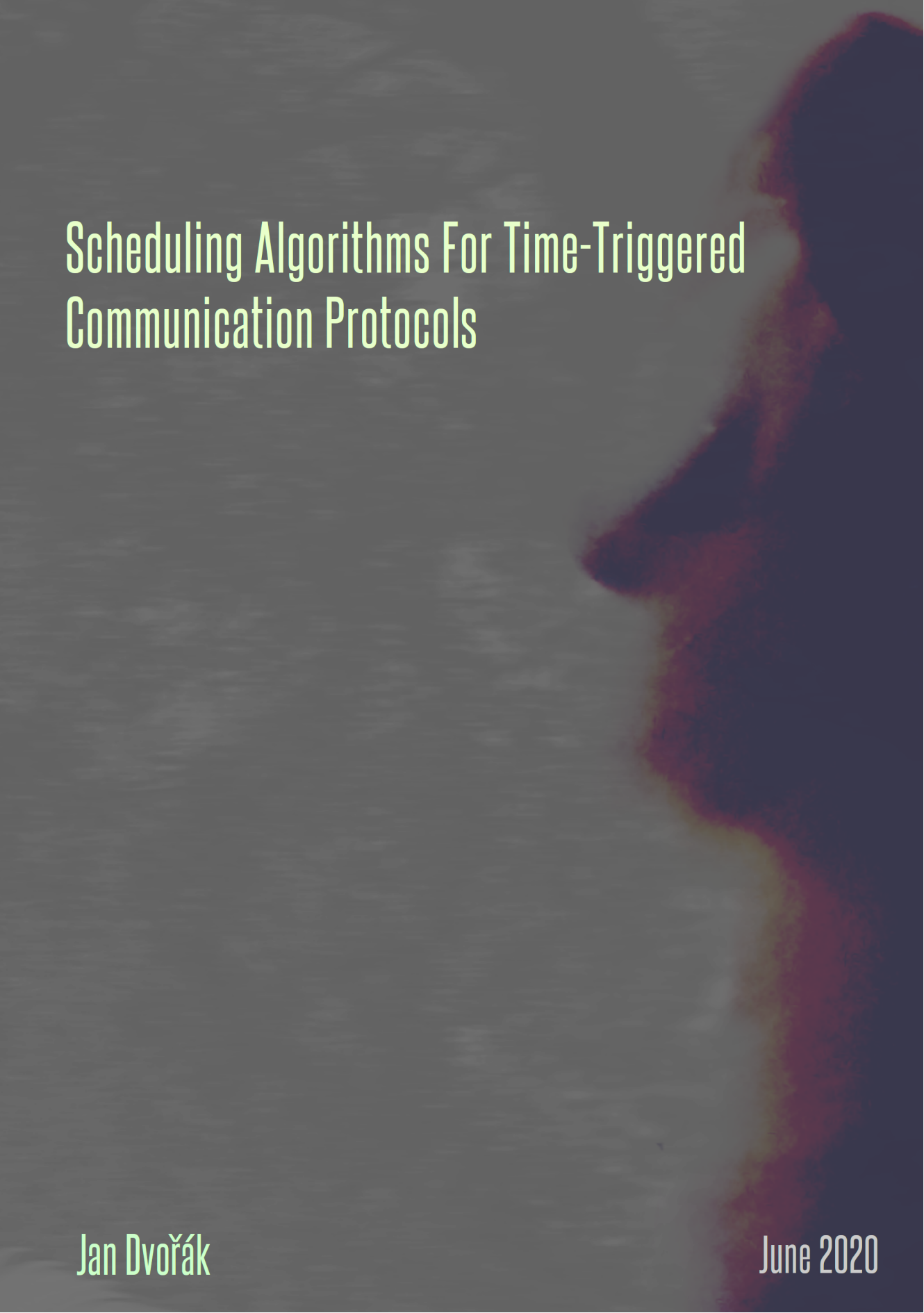

Scheduling Algorithms For Time-Triggered Communication Protocols

Jan Dvořák

June 2020

Scheduling Algorithms For Time-Triggered Communication Protocols

by

Jan Dvořák

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

Doctoral Thesis

Supervisor: Zdeněk Hanzálek.
Ph.D. programme: Electrical Engineering and Information Technology
Branch of study: Control Engineering and Robotics
Submission date: June 2020

Jan Dvořák: Scheduling Algorithms For Time-Triggered Communication Protocols, Ph.D. Thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Control Engineering, June 2020, Prague.

Acknowledgments

The long way to the Ph.D. study started a long time ago when I discovered my passion for technical disciplines. Thus, my special thanks belong to my father from whom I inherited the passion for them and who opened the world of natural science to me. Even if he died shortly before I started my Ph.D. study, he was a driving force for me all the time.

The Ph.D. study would be impossible for me without my outstanding supervisors. First of all, I express my thanks to Zdeněk Hanzálek, who always had an idea of how to improve the obtained results and how to present them to the readers better. Unfortunately, I was never able to make it as sexy as he wanted. Despite his busy schedule, he always finds a spot in his diary for me. Thanks to Petr Stuchlík, who was my supervisor in ŠKODA company. He showed me that science has two sides - the academic side and the industrial side - and that it is maybe the greatest challenge to find a bridge between them. I am also grateful to Přemek Šůcha who, even not being my official supervisor, was there to help me, and for his, maybe even more important, positive encouragement.

Man can not be satisfied with a job without a good team, and I have been extremely lucky and was part of an amazing collective. I thank my tablemate, Aasem Ahmad, who opened my mind to other cultures and who forced me to speak English every day. I am also grateful to Anna Minaeva (for showing me how far one can get if he is working hard), Libor Bukata (for his effort to master me in C++ and Linux), Roman Václavík (for his art of making a bright day even from a crappy one), Istvan Modos (for competing me in Ph.D. thesis writing and, thus, forcing me to make progress), Tonda Novák (who always make me describe everything as rigorous as I was able to), Zdeněk Bäumelt (for supervising my diploma thesis and opening the way to Ph.D. study), Svatava Petrachová (for helping me shake off all the administrative stuff), Michal Sojka, Pavel Píša and many others. During my Ph.D. study, I have met many students that influenced me. Among others, I am thankful to Martin Heller, who cooperated on the scheduling of the time-triggered communication in the TTEthernet network in his diploma thesis.

Personal life is as important as work life. Thus, I would like to thank to at least few of my friends Jan Hampl, Jiří Veselka, Petr Körner, Jonáš Kýček, Ondřej Hampl and Tomáš Kotek here too. They know how to set my mind at ease, even if circumstances are unfortunate.

Last but not least, I do want to express my gratitude to my family. Whole Ph.D. studies and this thesis would not be possible without their love and support. Namely, thanks to my wife Eliška, who pushed me to conclude the Ph.D. study as soon as possible and to my son Richard, who makes me postpone the thesis submission as far as possible. I am also very grateful to my mom and sister, who helped me to find some free time to finish the thesis and, together with my grandparents, supported me during my whole life.

Finally, I thank ŠKODA AUTO company, where I attended their Ph.D. program. This cooperation between university and industry made my narrow personal view much wider and helped me to distinguish between theoretical and practical problems.

Moreover, this thesis has been supported by the European Union's Horizon 2020

research and innovation program under grant agreement No. 688860 (HERCULES), the Grant Agency of the Czech Republic under the Project GACR P103/12/1994 and by the Grant Agency of the Czech Republic under the Project FOREST GACR P103-16-23509S.

Jan Dvořák
Prague, June 2020

Declaration

This doctoral thesis is submitted in partial fulfillment of the requirements for the degree of doctor (Ph.D.). The work submitted in this thesis is the result of my own investigation, except where otherwise stated. I declare that I worked out this thesis independently and I quoted all used sources of information in accord with Methodical instructions about ethical principles for writing academic thesis. Moreover, I declare that it has not already been accepted for any degree and is also not being concurrently submitted for any other degree.

Jan Dvořák
Prague, June 2020

Abstract

Modern automotive systems present a number of new technological challenges for developers and researchers in the field of electronic control systems. There appear efforts to replace the originally mechanical, hydraulic, or pneumatic components with electronic systems. Steer-by-wire, break-by-wire, or x-by-wire systems are examples of emerging technologies, where the original mechanical connection is replaced by an electrical or electromechanical connection. However, such an electrical connection puts high demands on the reliability and time-determinism of the entire system, an important part of which is the communication system. Moreover, the requirement to use a shared communication system for the whole system, which leads to cost savings and thus better competitiveness of the final product, is also frequent. The problem becomes even more complicated as the number of assistance systems in cars and, consequently, the units that need to be interconnected is increasing. In addition, the approaching era of vehicles with autopilot features requires high volume demanding communication, such as signals from cameras or lidars, be part of the control loop of the system. Therefore, they have to meet high reliability too. As a result, modern automotive systems require communication systems that are able to transfer large volumes of data reliably and deterministically. Time-triggered communication protocols such as FlexRay or TTEthernet have been designed for this purpose. However, their reliability and efficiency are closely tied to the quality of the schedule that communication follows.

This work deals with the problem of scheduling time-triggered communication on FlexRay and TTEthernet protocols. The first part of the work focuses on the problem of communication scheduling of FlexRay, which provides two communication channels that can be used independently. Since electronic units can be connected to only one channel, the scheduling method aims to determine which channel the unit will be connected to, so that the resulting communication schedule is as efficient as possible. The second part is also focused on creating schedules for FlexRay. Here, however, the issue of scheduling in the development process, where it is necessary to take into account several variants of the product and also backward compatibility with previous products, is studied. An example of such a development process can be found in the automotive industry, where multiple variants of a car are created on a common platform, and they are required to be compatible with each other. The third part of the study is focused on scheduling time-triggered communication in the TTEthernet network. In compliance with the second part of the study, it is required to maintain backward compatibility with its predecessor.

Each part of the study includes an analysis of related works and a formal definition of the problem to be solved. Subsequently, the proposed algorithm that solves the defined problem is described. The algorithm is then verified from the qualitative and quantitative point of view, and the phenomena of given specific problems that are reflected in the resulting schedules are studied in experiments.

Keywords: time-triggered communication scheduling, combinatorial optimization, real-time systems

Abstrakt

Moderní automobilové systémy přinášejí řadu technologických výzev pro vývojáře elektronických řídicích systémů. Objevují se snahy nahradit původně mechanické, hydraulické či pneumatické komponenty elektronickými systémy. Příkladem toho mohou být třeba systémy steer-by-wire, break-by-wire či x-by-wire, kdy původně mechanické propojení je nahrazeno elektrickým či elektromechanickým propojením. Takovéto elektrické propojení však klade vysoké nároky na spolehlivost a determiničnost celého systému, jehož důležitou částí je komunikační systém. Častý je navíc i požadavek využití společného komunikačního systému pro celý systém, což vede k finančním úsporám a tím i lepší konkurenceschopnosti finálního produktu. Vše se stává ještě komplikovanějším problémem ve chvíli, kdy se rozšiřuje počet asistenčních systémů v automobilech a tedy i jednotek které je mezi sebou potřeba propojit. Blížící se éra vozidel s prvky autonomního řízení navíc vyžaduje, aby i komunikace spotřebovávající značnou část přenosového pásma, jako jsou třeba signály z kamer či lidarů, byla součástí řídicího systému a tudíž je nutné splnit vysokou spolehlivost též. Ve výsledku to znamená, že moderní automobilové systémy vyžadují komunikační systémy, které jsou schopny přenést velké objemy dat spolehlivě a deterministicky. Pro tento účel byly navrženy komunikační protokoly řízené časem jako je FlexRay či TTEthernet. Jejich spolehlivost a přenosové vlastnosti jsou však úzce svázány s kvalitou rozvrhu, dle kterého se komunikace řídí.

Tato práce se věnuje problému vytváření rozvrhů pro komunikační protokoly FlexRay a TTEthernet. První část se zaměřuje na problém vytváření rozvrhů pro FlexRay, který poskytuje dva komunikační kanály, které mohou být využity nezávisle. Jelikož mohou být elektronické jednotky připojeny jen k jednomu kanálu, snadží se metoda určit ke kterému kanálu bude jednotka připojena tak, aby byl výsledný rozvrh komunikace co nejefektivnější. Druhá část práce je také zaměřena na vytváření rozvrhů pro FlexRay. Zde je však studována problematika vytváření rozvrhů ve vývojovém procesu, kdy je potřeba brát v potaz více variant produktu najednou a navíc i zpětnou kompatibilitu s předešlými produkty. Příklad takového vývojového procesu můžeme najít v automobilovém průmyslu, kdy více variant automobilu je vytvářeno na společné platformně a je požadováno, aby byly mezi sebou kompatibilní. Poslední, třetí část studie, je zaměřena na rozvrhování komunikace řízené časem v síti TTEthernet. Stejně jako v druhé části i zde je požadováno zachování zpětná kompatibility s předešlou verzí produktu.

Pro každou část studie je vypracována rešerše s rozborem příbuzných vědeckých prací a formálně nadefinován problém, který daná část řeší. Následně je popsán navržený algoritmus, který nadefinovaný problém řeší. Každý algoritmus je pak ověřen z pohledu kvalitativního i kvantitativního a na experimentech jsou studovány jevy daných specifických problémů, který se projevují na výsledných rozvrzích.

Klíčová slova: rozvrhování komunikace, kombinatorická optimalizace, systémy reálného času

Goals and Objectives

The thesis should focus on scheduling safety-relevant time-triggered communication on modern communication systems. The main goals, the thesis should achieve, are the following:

- Examine literature, papers, and specifications related to the time-triggered communication systems and find open challenges and possible improvements in the scheduling area. Focus on the problems that obstruct introducing time-triggered communication in the industrial sector.
- Deduce a formal description of the discovered problems. Formulate the problem constraints and objectives while considering the safety-relevant and time-deterministic behavior.
- Develop exact or heuristic algorithms that can solve the scheduling problems for industrial-sized instances. The schedules have to be obtained in a reasonable time for the product development process.
- Verify the developed algorithms by experiments. Discuss the obtained results from the quality point of view as well as from the time complexity and scalability point of view. Investigate the uncommon properties of the stated problems and the proposed algorithms.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

We live in a period of the rapid development of new technologies. Technology progress became a part of our lives and allowed us to live in comfort. Informatics and electronics are among the fields which evolved most in the last decades. Electronic systems with relatively simple functionality were huge and heavy devices fifty years ago. The systems with much more complex functionality can be wearables today. This advancement influenced even industries, e.g., automotive, avionic, or health care, that tends to be somewhat conservative. The products of these industries interact closely with humans, and any failure of a system can cause jeopardy of life. Thus, it is necessary to be able to prove that the system behaves as expected and to minimize the probability of system failure. The concrete example is a trend in the automotive industry where vehicles with autonomous driving are expected to appear in series production in the second half of the next decade. The logic of the autonomous driving control system needs to fuse different dataflows, as signals from the camera, lidar, and other sensors, process them, and deduce the commands for steering, braking, etc. To perform this task in a way that the safety of passengers is ensured puts demands on system determinism, real-time processing, fault-tolerance, etc. It is a great challenge to satisfy all these demands.

1.1 Communication systems

Complex systems are often distributed among several control units, sensors, and actuators. All these communication nodes need to be interconnected by a communication system that allows the exchange of information among them. Thus, the communication system is a crucial part of the whole system, and its reliability is strongly dependent on the reliability of the interconnection.

The communication systems can be categorized into two groups - Event-Triggered and Time-Triggered. The approaches differ in the way the message with the information is triggered to be transmitted into the communication system. Transmission to the ET communication system is decided at run-time. Thus, the message can be transmitted whenever some event occurs. These transmissions can be triggered by an external event, e.g., change in the state of the environment, or by an internal event, as timer timeout, for example. The main advantage of this approach is its flexibility. The communication system can accommodate new signals or nodes without the necessity of the system redesign. Moreover, communication is utilized according to actual demands, which reduces unnecessary overhead. However, the principle of the triggering makes the communication system to be nondeterministic. In the case of a safety-critical control system, proving the correct worst-case functionality of the communication system is necessary. The worst-case analysis can ensure the correct behavior, but it also reduces all the advances of the ET.

On the other hand, the transmission of a message to the TT communication system is triggered according to the schedule, which is determined in the system design time. Together with the time synchronization among nodes, which is

essential here, TT communication systems ensure time-deterministic information exchange among nodes in the system. Thus, the reliability and easy verification of the correct function of the communication system is obtained at the price of flexibility loss. These benefits are essential for safety-critical systems where the use of TT communication makes the certification process significantly shorter and less expensive. However, the reliability of the whole TT system is dependent on the quality of the schedule, which the communication follows. Thus, the responsibility of the satisfying, for example, the timing constraints, is moved to the schedule designer. To create a feasible schedule becomes a challenging task with the increasing number of messages or requirements.

Modern communication systems as FlexRay, TTEthernet, or Ethernet with 802.1Qbv aim to take advantage of both principles and combine TT communication and ET communication into one communication protocol. In that case, the TT communication follows the schedule while transmission of the ET communication can occur only in a time when no TT communication transmission is scheduled. The use of both principles at once helps to keep the certification of high-critical systems simple and also to keep flexibility for low-critical systems.

1.2 Variant management

The problem of the schedule creation becomes even more complicated in the real development process. The manufacturer often develops and creates several products that share the same components. If the schedules for the communication systems of each safety-critical product are built from scratch, it is needed to certificate each one individually, which results in extra costs. Thus, it is beneficial to consider the variant management during the product design phase and create such a schedule that can be shared among the resulting variants of the product. The variant management is even more critical in the industry with the after-sale market where you would need to adapt the schedule in the component (e.g., parking camera in a vehicle) according to product version in which it should be replaced. Without a shared schedule, the component/modular systems would lose their benefits as the product variants, and their components would be unmanageable.

Moreover, the products develop in time. To keep the principles of variant management valid even in this case, the new product variants should be built incrementally, and the backward compatibility with its preceding variants must be satisfied as much as possible.

1.3 Objectives and constraints of scheduling

As modern communication systems use ET and TT altogether, it is not only necessary to find a feasible schedule that satisfies all the constraints. The schedule has to be compact as much as possible to keep sufficient bandwidth for ET communication. Moreover, there has to be a free capacity in the communication system kept for future product variants. Without that free capacity, it would not be possible to incorporate new devices and messages to the communication system and to keep

backward compatibility. Thus, the main objective is to find the most compact schedule that satisfies all the real-time and application requirements.

1.4 Contributions of the thesis

The work of the thesis is focused explicitly on scheduling TT communication in two communication systems - the FlexRay bus and the TTEthernet network. The main contributions of the thesis in the field of scheduling TT communication on FlexRay bus are:

1. Provided idea of utilizing both independent FlexRay communication channels based on the classification of signals into two groups: signals that do require and do not require fault-tolerance [Sec. 2.2]
2. Formal formulation of the TT communication scheduling problem in the FlexRay bus that allows utilizing both communication channels [Sec. 2.2]
3. The NP-hardness of the stated scheduling problem is proved [Sec. 2.3.1]
4. The heuristic algorithm with the problem decomposition to the ECU-to-channel assignment subproblem and channel scheduling subproblem is presented [Sec. 2.3] and includes
    - The MILP formulation for the exact solution of the ECU-to-channel assignment subproblem [Sec. 2.3.1]
    - The heuristic method for solving the ECU-to-channel assignment subproblem for big instances [Sec. 2.3.1]
    - The heuristic method for channel scheduling subproblem [Sec. 2.3.2]
5. Evaluation of the scheduling algorithm from objective function (number of allocated slots) point of view and computation complexity point of view, and comparison of provided the ECU-to-channel assignment methods with the alternative method based on genetic algorithm [Sec. 2.4]
6. Provided idea of considering the practical product development process consisting of multi-variant management and incremental development in the FlexRay TT communication scheduling procedure [Sec. 3.1.1]
7. Formal formulation of the incremental multi-variant FlexRay static segment scheduling problem with real-time constraints [Sec. 3.2]
8. The incremental and multi-variant heuristic scheduling algorithm [Sec. 3.3], which includes:
    - The exact algorithm for resolving the conflicts in the original schedule while minimizing the number of changes [Sec. 3.3.4]
    - The signal scheduling algorithm based on First-Fit Decreasing method for Bin-Packing problem [Sec. 3.3.4]
    - A new extensibility optimization method, which can adapt to a particular input instance by utilizing the probability distribution of the signal parameters [Sec. 3.3.4]
    - The graph coloring formulation of slot scheduling sub-problem and its convenient ILP formulation [Sec. 3.3.4]
9. Examination and discussion of the impact of multi-variant and incremental essence on scheduling [Sec. 3.4.2, Sec. 3.4.3]

10. Evaluation of the algorithm on the sets of both synthetic and real-case inspired instances and the significance of the extensibility optimization [Sec. 3.4.1, Sec. 3.4.4, Sec. 3.4.5]
11. Evaluation and discussion on how basic network configuration parameters influence the bandwidth efficiency of the resulting schedules [Sec. 3.4.6]
12. Verification of the resulting schedules on the FlexRay powered system [Sec. 3.4.7]

The main contributions in the field of scheduling TT communication on TTEthernet network are:

1. Provided idea of considering backward compatibility in the TTEthernet scheduling process [Sec. 4.2]
2. Use of TT communication schedule makespan as the criterial function (which follows the practice from FlexRay bus) [Sec. 4.3.6]
3. Formal formulation of the incremental TT communication scheduling problem on TTEthernet, with real-time constraints. [Sec. 4.3]
4. The three-stage heuristic scheduling algorithm [Sec. 4.4], which includes:
   - The routing algorithm that balances the communication load among links [Sec. 4.4.1]
   - The message-to-integration cycle assignment algorithm that balances the communication load among integration cycles [Sec. 4.4.2]
   - The message scheduling method based on RCPSP model of the problem [Sec. 4.4.3]
5. Examination and discussion of the impact of incremental essence on the TTEthernet scheduling [Sec. 4.5.2, Sec. 4.5.3]
6. Evaluation of the proposed algorithm from the quality and the performance point of view [Sec. 4.5]

1.5 Outline of the thesis

The rest of the thesis is organized as follows. Chapter 2 studies the problem of efficient use of both communication schedules provided by the FlexRay bus. The study utilizes the fact that not all messages and ECUs are included in the fault-tolerant application and, thus, are not necessarily transmitted into both channels. Moreover, if an ECU is not part of a fault-tolerant application, it can be connected to the FlexRay bus only through one port. The heuristic scheduling algorithm is provided for this problem, and it is evaluated on real-case and synthetic datasets.

The constraints of the multi-variant management and the incremental development process are introduced to the FlexRay TT communication scheduling problem in Chapter 3. The multi-variant management demands to create schedules for a set of product variants at once such that the shared signals are scheduled in the same position. On the other side, the incremental development process demands to keep the backward compatibility of the new product variants with their predecessors. The problem is solved by the heuristic algorithm that is able to create schedules even for instances of industrial size is described and evaluated. Finally, the essence of the multi-variant and incremental scheduling on the creation of FlexRay TT communication schedules is examined and discussed.

Chapter 4 focuses on the problem of creating schedules for TT communication in the TTEthernet network while considering the backward compatibility. The problem of creation of the TT communication schedules on TTEthernet is even more challenging than in the case of FlexRay because the simple bus topology is replaced by the switched network topology. Thus, not only one schedule is needed for the whole communication system, but there must be created the schedule for each link in the network, and, moreover, the message routing needs to be decided. The MILP and CP based heuristic algorithm are described to solve the problem. The algorithm is examined in detail for its scalability evaluation, and its usability for the industrial size instances is discussed.

Chapters 2, 3 and 4 corresponds to individual journal papers and each one tackles different problems. Thus, the chapters are self-contained and can be read separately.

Chapter 5 concludes the thesis, discusses the achieved results, and the topics left open for future work.

Chapter

2 FlexRay Static Segment Scheduling on Two Independent Channels with Gateway

The FlexRay bus is a communication standard used in the automotive industry. It offers a time-deterministic message transmission in the static segment following a time-triggered schedule. Even if its bandwidth is ten times higher than the bandwidth of CAN, its throughput limits are going to be reached in high-class car models soon. A solution that could postpone this problem is to use an efficient scheduling algorithm that exploits both channels of the FlexRay. The significant and often neglected feature that can theoretically double the bandwidth is the possibility to use two independent communication channels that can intercommunicate through the gateway.

In this chapter, we propose a heuristic algorithm that decomposes the scheduling problem to the ECU-to-channel assignment subproblem which decides which channel the ECUs (Electronic Control Units) should be connected to and the channel scheduling subproblem which creates static segment communication schedules for both channels. The algorithm is able to create a schedule for cases where channels are configured in the independent mode as well as in the fault-tolerant mode or in cases where just part of the signals are fault-tolerant. Finally, the algorithm is evaluated on real data and synthesized data, and the relation between the portion of fault-tolerant signals and the number of allocated slots is presented.

2.1 Introduction

Nowadays, the automotive industry is evolving fast. Modern vehicles consist of many critical systems, like powertrain and chassis control or advanced driver assistance systems. There is also a huge effort to supplant obsolete mechanic and hydraulic control systems by electronic systems. Consequently, the number of latest vehicle models will aim to use x-by-wire systems (already used in Nissan Infinity Q50 for example) shortly. This trend causes an increase in the number of ECUs and also in the number of messages that have to be exchanged among these units. Therefore, it is hard for conventional communication buses, such as the CAN bus, to follow this trend. The FlexRay bus has been designed to satisfy such a demand as it is well suited for real-time and safety-related applications and provides transmission rates up to 10Mb/s. Its static segment with the time division multiple access (TDMA) scheme can handle real-time requirements, while the message transmission follows a given schedule. The interconnection with other buses (e.g. CAN) is usually done via a gateway node.

2.1.1 Motivation

In practice, the FlexRay standard has been used just for a few years, but its limits could be reached soon if we do not take advantage of all the opportunities it offers. This problem currently occurs in premium class vehicles because they contain a lot of advanced driver assistance systems that need a generous bandwidth. For example, data from an intelligent camera become subject to safety requirements. The same also holds for lidar, radar, ultrasonic and other signals which require deterministic processing. Some signals need to fulfill the fault-tolerant requirements while others just need to be transmitted deterministically. The problem with a lack of bandwidth is often solved by splitting the whole network into separate buses which are interconnected by gateways. Unfortunately, this solution causes synchronization problems when real-time constrained messages have to be exchanged between different buses. It is also economically inconvenient because an additional infrastructure involves extra costs. The Automotive Ethernet [6,31] could introduce the desired bandwidth, but it does not provide the safety by design in its 2nd generation. Thus, the Ethernet is still more suitable for infotainment than safety-related applications today.

One way to efficiently use the bandwidth provided by the FlexRay bus is to create an efficient schedule for the TDMA part. Another way, unique to the FlexRay standard, is to use the FlexRay channels independently. Despite that it can theoretically multiply the transmission rate by two, this property is usually overlooked by scheduling algorithms.

In this chapter, we combine both ways to minimize the number of slots used by the periodic message transmission in the static segment and, consequently, to save the bandwidth for the dynamic segment.

2.1.2 FlexRay overview

The FlexRay bus, standardized in ISO 17458 - FlexRay communications systems [29], is a modern bus that has been developed to satisfy the performance and safety requirements of the advanced driver assistance systems. This is often coupled with the AUTOSAR Specification [4,5] in the automotive industry. The FlexRay communication is organized in cycles. Each communication *cycle* has its own six-bit identifier, denoted as *cycleID*, and there can be up to 64 cycles. The sequence of these cycles is denoted as a hyperperiod and is periodically repeated. An example of the communication scheme on the FlexRay bus is depicted in Fig. 2.1, wherein the hyperperiod consists of four communication cycles. Each communication cycle contains four segments:

- Static segment
- Dynamic segment
- Symbol window (SW)
- Network idle time (NIT)

The time-critical signals are exchanged, using a time-triggered scheme, based on time division multiple access in the static segment. The dynamic segment fulfills the requirements of event-triggered communication. The other two segments, SW and

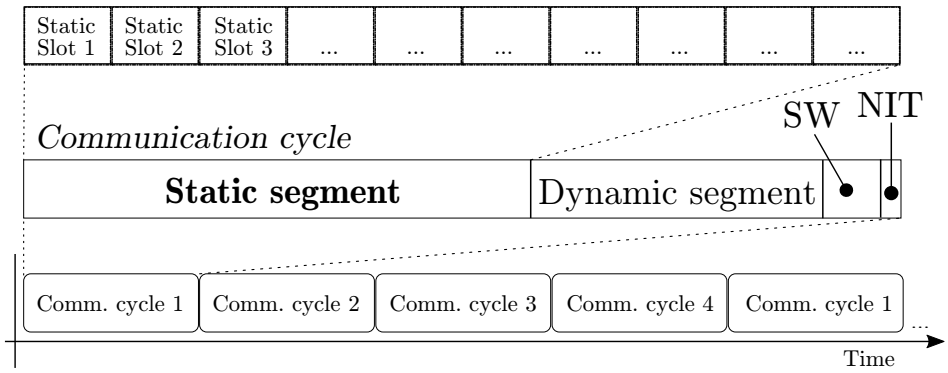

Figure 2.1: FlexRay communication scheme

NIT, are used for network management and inner clock synchronization. Among all these, only the static segment and NIT are mandatory.

This chapter deals only with the time-triggered communication in the static segment. The static segment is divided into time intervals of same duration, called static slots (referred to as just *slots*, hereafter). A given slot is reserved for a given ECU (i.e., the frames transmitted to a given slot need to be from the same ECU in all cycles)[1]. The data structure used by the ECUs in transmitting the data is called a frame. Each frame is identified by its cycle and slot number. The frame can contain more than one signal, but the sum of payloads of these signals must not exceed the duration of the slot. The signals packed into the same frame can be distinguished by the offset in the frame. The schedule determines in which instant the frame is transmitted to the network.

The bus offers two channels for communication: channel A and channel B. An ECU can be connected to both or just to one of them. At least two ECUs, called synchronization ECUs, must be common to both channels. The communication can operate in two modes from the channels point of view: in the independent mode (when the communication on channel A is independent of the communication on channel B) or in the fault-tolerant mode (when the communication on channel B is synchronized with the transmission on channel A). The fault-tolerant mode is beneficial for error detection. However, fault-tolerance is usually not necessary for all the signals and in these cases the utilization of independent channels can be an efficient way to save the bandwidth.

In this chapter, we focus on the assignment of ECUs to two independent channels and the scheduling of signals to a particular cycle, slot and offset in the frame for both channels.

[1]The FlexRay standard 3.0 and later allow different nodes to transmit frames within the same slot, on the same channel, in different communication cycles. However, this feature was not used in this study, because it requires different scheduling model which introduces extra complications with variant management in subsequent incremental scheduling iterations, defeating the idea of keeping the schedules easily manageable even in later scheduling iterations without breaking backward compatibility.

2.1.3 Related works

A significant effort was made to find bandwidth saving and safety-related constraints satisfying communication schemes for the FlexRay protocol over the last six years. The FlexRay communications system is described in detail in ISO 17458 [29]. In the automotive industry, this bus is often coupled with the AUTOSAR Specification [4,5] which extends the FlexRay standard by new safety-related constraints. Nowadays, BMW, Audi, Mercedes-Benz, etc. use the FlexRay bus in several series-production vehicles.

A milestone in the static segment scheduling area is the work of Lukasiewycz et al. [39] where the transformation of the fundamental static segment scheduling problem to a specific two-dimensional bin packing problem was introduced. The authors presented an ILP model and also a successful heuristic based on the first fit policy for the bin packing problem. The objective is to minimize the number of scheduled slots and to obtain an extensible schedule. A similar problem extended by time constraints was proposed by Hanzalek et al. [26]. This work employs the idea of a two-stage heuristic, where, in the first step, the frame packing is performed and, in the second step, the schedule of time-constrained frames is obtained. Kang et al. suggested another frame packing algorithm in [32] where the best fit decreasing heuristic and the ILP model were utilized. However, the paper minimizes the number of used frames instead of the number of allocated slots and the period of signals can be an arbitrary multiple of the communication cycle. Tanasa et al. used the CLP formulation to strengthen the system reliability by the repetitive transmission of more critical signals in [63]. In [19], we proposed a heuristic for the time-constrained static segment scheduling problem that takes more vehicle variants into account and creates a multi-schedule for all of them at once.

Lange et al. [36] used the rate monotonic scheduling method for the response time analysis of the static segment. However, this paper requires modifications of the middleware. Bouhouch et al. described an analysis of Data Distribution Service (DDS) for the FlexRay bus based on the subscription-publication paradigm in [8] and Sojka et al. [55] considered flexible reservation mechanisms for distributed real-time applications. The concept of time analysis considering both static and dynamic segment of the FlexRay communication protocol is presented in [28] and [42].

The methods described in the previous papers consider the channels to be set into the fault-tolerant mode. Thus, communication is duplicated even for signals that do not require the fault-tolerant scheduling and the potential to save the bandwidth is wasted.

It is necessary to decide, for each ECU, if the ECU should be connected to channel A, B or to both of them to divide the communication between the channels. A similar problem from computer science is clustering. Some widespread clustering methods are expectation-maximization (EM) and the K-Means algorithm [22]. Graph clustering and spectral clustering methods [24] are important for clustering problems that can be modeled by graphs. Another classical combinatorial optimization problem related to assigning items to subsets is number partitioning [43].

2.1.4 Outline of the chapter

The rest of this chapter is organized as follows: Section 2.2 describes the FlexRay static segment scheduling problem for two independent channels with a gateway. In Section 2.3, the NP-hardness of the problem is proved, and the heuristic algorithm with the problem decomposition to the ECU-to-channel assignment subproblem (solved by exact and heuristic method) and channel scheduling problem (solved by heuristic method) is presented. A computational efficiency and a performance evaluation of the proposed approach are presented in Section 2.4. Section 2.5 concludes the chapter.

2.2 Problem statement

The problem is to create FlexRay static segment schedules for independent channels that can intercommunicate via a gateway. Such a model is used in cases where fault-tolerance is not critical (an undetected loss of one signal instance cannot cause a jeopardy) for all the signals. Our aim is to find a schedule that minimizes the number of allocated slots and, consequently, reduces the length of the static segment in the communication cycle as much as possible.

The configuration of the FlexRay network contains many parameters, which are not directly related to the schedule optimization process, such as the duration of the communication cycle L, the payload of the static slot, etc. We assume that these parameters are given by network designers, and they follow the specification of the manufacturer.

The set of ECUs $\mathcal{N}$ consists of three disjoint subsets $\mathcal{N} = N \cup N^{\mathrm{GW}} \cup N^{\mathrm{Comm}}$, where N is the set of one port ECUs. These ECUs can be connected either to channel A or channel B but not to both of them. An ECU connected to one channel may need to receive data from the second channel. A special gateway ECU N^{GW} serves as a mediator for such a data exchange between channels. The gateway has no own data to transmit. It just receives data from one channel and sends them to the second one. The gateway can interconnect the FlexRay bus with the CAN bus in practice, but it is not the subject of interest in this work. There can be more than one gateway ECU to decrease the impact of the single point of failure problem when the gateway ECU is malfunctioning. This issue is tackled in [53]. However, it is assumed for the sake of simplicity that there is just one gateway ECU in the N^{GW} set in the rest of the chapter. N^{Comm} represents the set of common ECUs. These ECUs are connected to both channels. According to the FlexRay standard, the minimal number of common ECUs is two because at least two ECUs have to be used to synchronize the network. The common ECUs can transmit their data to both channels, but they are not allowed to transfer any data between channels. The assignment of the ECUs to the subsets of $\mathcal{N}$ is given.

The data that have to be exchanged in the network are represented by a signal set S. Each signal s_i from the set S has the following parameters:

q_i - unique identifier of the ECU which transmits s_i
p_i - the signal period
c_i - signal length/payload in bits
r_i - release date
d_i - deadline
f_i - fault-tolerance of the signal
Q_i - the set of signal receivers

The q_i identifier of a signal can be any ECU from N or N^{Comm} but it cannot be N^{GW}. The signal s_i is assumed to be transmitted only once in the FlexRay cycle. Its period p_i must be a multiple of the cycle duration L, and no jitter is allowed. Furthermore, according to the AUTOSAR specification, the period must be equal to L multiplied by some power of two (i.e. $p_i = \{h \cdot 2^n \mid n = 1 \dots 6\}$). The payload of the signal c_i represents the data payload. The fragmentation of signals is not allowed. Signals can be packed to be transmitted in one frame, but the sum of their payloads must not exceed the static slot payload. The fault-tolerance f_i of the signal is equal to 1 if the signal s_i has to be transmitted to the both channels at once otherwise it is equal to 0. Note that if $f_i = 1$ then $q_i \in N^{\mathrm{Comm}}$ otherwise it would not be possible to transmit the signal to both channels by ECU q_i. The signal receivers set Q_i contains identifiers of all ECUs that need to receive the signal. If a receiver is in a different channel than the transmitter and $q_i \in N$ then the gateway has to receive the signal from the channel with the transmitting ECU and forward it to the second channel. The signals transmitted by the gateway to the second channel are called signal images, and we denote them by s'_i.

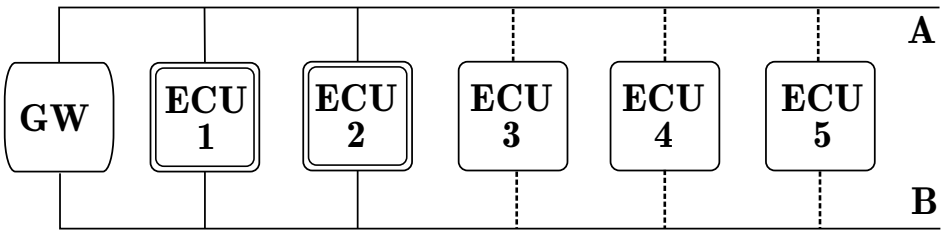

Figure 2.2: FlexRay network with two independent channels

The goal is to find an assignment $s_i \rightarrow [b_i, y_i, t_i, o_i]$, where b_i represents the channel to which the signal is transmitted, y_i is the identifier of the first signal occurrence communication cycle (cycleID) in the schedule, t_i is the identifier of the slot (slotID) and o_i is the offset of the signal in the frame and, furthermore, to find an assignment $s'_i \rightarrow [b'_i, y'_i, t'_i, o'_i]$ for images of signals that have any receiver connected to a different channel than the transmitter. Note that we can deduce the position of all signal occurrences in S and S' (the set of signal images) from this assignment because signals have no jitter. No two signals are tolerated to be overlapped in the schedule for a particular channel. Therefore, it is necessary to know which channel the ECU from N is connected to. Consider the example shown in Fig. 2.2. The common ECUs N^{Comm} are highlighted by double borders. The ECU labeled as GW is the gateway ECU N^{GW}. The ECUs from N^{Comm} and N^{GW}

are always connected to both channels. The assignment of the ECUs from N to a channel is unknown (indicated by a dotted line in Fig. 2.2) and it is the subject of the optimization. Our aim is to find a feasible assignment in such a way that the maximal identifier of any used slot is minimal.

2.2.1 Example 1: Simple case with two cycles and ten signals

We introduce a simple example for a better understanding of the problem statement. The infrastructure presented in Fig. 2.2 is used. The duration of the communication cycle $L = 1$ ms and slot payload 8 bytes is assumed. The hyperperiod consists of two cycles. There are ten signals s_1 ... s_{10} with the following parameters:
$q_i = [1, 2, 2, 2, 3, 3, 4, 5, 5, 4]$,
$p_i = [1, 2, 2, 2, 2, 1, 1, 1, 2, 2]$ in ms,
$c_i = [8, 4, 8, 8, 4, 4, 4, 4, 4, 4]$ in bytes,
$f_i = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]$,
$Q_i = [\{2, 3\}, \{4, 5\}, \{4\}, \{5\}, \{4, 5\}, \{4, 5\}, \{3, 5\}, \{2\}, \{3, 4\}, \{3\}]$.
The release date $r_i = 0$ ms and deadline $d_i = 2$ ms for all the signals for the sake of simplicity. One optimal solution is shown in Fig. 2.3.

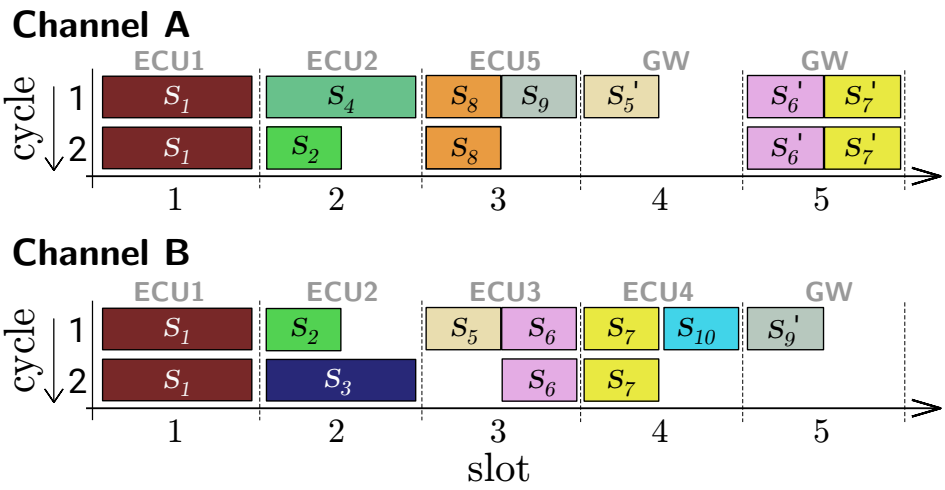

Figure 2.3: Feasible schedules for Example 1

The schedule for channel A is shown at the top of Fig. 2.3 and the schedule for channel B is at the bottom. The rows of the schedule for the given channel represent the cycles, and the columns represent the slots. For example, the signals s_8 and s_9 (its first occurrence) are packed in the same frame scheduled in the third slot of the first cycle in channel A. The signal s_8 has the zero offset, and the signal s_9 has an offset of 4 bytes. It is denoted as $s_8 \to [A, 1, 3, 0]$ and $s_3 \to [B, 2, 2, 0]$. The remaining occurrences of a signal s_8 can be deduced from its period ($p_8 = 1$ ms). The signal images are marked by an apostrophe (e.g. signal image of s_5 is s_5'). The pale labels above the columns determine which ECU operates in the given slot. Please notice that even though signals s_6 and s_7 are transmitted by different ECUs

(ECU 3 and ECU 4 respectively), their images s'_6 and s'_7 can be transmitted in the same slot.

From Fig. 2.3, the reader can derive that ECU 3 and ECU 4 are connected to channel B, ECU 5 to channel A and ECUs 1, 2 and GW are allowed to transmit signals to both channels.

2.3 Algorithm

The design of the proposed algorithm is explained in this section. The problem is very complex because even the channel, which the individual ECUs from N are connected to, is unknown. Solving an industrial-sized problem by exact methods would result in an unacceptable computation time. Thus, a heuristic algorithm depicted in Algorithm 2.1 is used.

Algorithm 2.1 The pseudocode for the iterative static segment scheduling algorithm

Input: S, $\mathcal{N}$, α
Output: Best found schedule for FlexRay static segment

```
ChannelAssignment Asg
Schedule Schd, bestSchd
β ← 1
repeat
    Asg ← ECU-TO-CHANNEL(S, N, α, β)
    Schd ← CHANNELSCHEDULING(S, Asg)
    β ← sqrt( max_{s∈S_A∪S'_A} t_s / max_{s∈S_B∪S'_B} t_s )
    bestSchd ← GETBETTER(Schd, bestSchd)
until ISSTOPCONDITIONMET(Asg);
return bestSchd
```

The iterative algorithm is divided into three phases. In the first phase, the ECUs from N are assigned to the channels and the schedules for channels A and B are created in the second phase. The last phase updates coefficient β.

The coefficients α and β are modifiers of the ECU-to-channel assignment criterion function. α is the weight of the gateway throughput penalization and it can be determined by a network designer. β outbalances the payload of the individual channels. An optimal result of the first phase does not ensure an optimal result of the channel scheduling phase. The balanced ECU-to-channel assignment (the assignment where the payload in both channels is almost equal) can still result in a schedule with significantly more slots occupied in one channel than in another. Thus, the β coefficient is recalculated to counterweight this imbalance in the next iteration. The new value of β is equal to the square root of the ratio between the number of slots allocated in channel A to the number of slots allocated in channel B. The actual schedule *Schd* (representing schedule for channel A and B together here) is compared with the best schedule already found *bestSchd* at the end of each iteration. The one that needs smaller number of allocated slots is stored. The stop

condition of the iterative algorithm is met if the number of iterations exceeds a threshold or if the cycling of the algorithm is detected (the actual assignment was already found in the past).

Best found schedule *bestSchd* is returned in the form of a FIBEX [3] database at the end. The FIBEX file allows direct loading of resulting schedules to tools often used in the automotive industry (such as Vector CANoe).

2.3.1 ECU-to-channel assignment

Each ECU from N is assigned to a particular channel at the ECU-to-channel assignment phase. Our aim is to find such an assignment which seems to be promising for finding a good schedule in the second phase. It is assumed that if there is a smaller data payload to be transmitted in a channel, then the resulting schedule of the channel will be shorter. According to that assumption, the task is to find such an assignment that minimizes the number of bits transmitted in any channel.

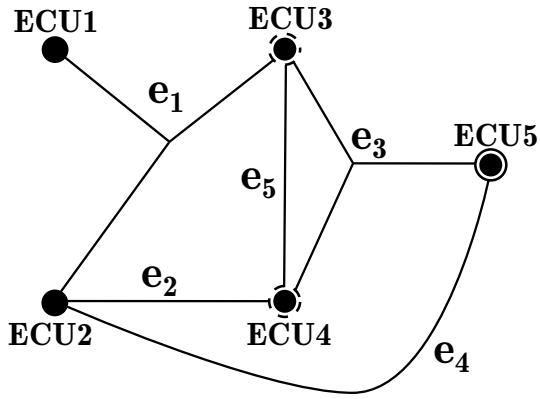

Figure 2.4: Hypergraph for Example 1

The problem can be modeled by a hypergraph. Fig. 2.4 depicts the example of a hypergraph resulting from the data in Example 1. Each vertex represents one ECU from the sets N and N^{Comm}. The vertices are connected by hyperedges. One hyperedge represents an aggregated set of signals with the same endpoints. The endpoints are receivers and the transmitter of the signal. The signals s_5, s_6, s_7, s_9 in Example 1 can be aggregated to one hyperedge e_3 because their set of the endpoints is the same: $\{3, 4, 5\}$. A payload of the hyperedge w_e is the sum of its signal payloads. It is not necessary to distinguish between the transmitter and the receiver here.

The task is to mark vertices which represent the ECUs from N. The marking corresponds to their assignment to the channels. The ECUs connected to both channels are not outlined (ECU 1 and 2). The ECUs assigned to channel A have a solid black outline (ECU 5) and the ECUs with a dashed outline are assigned to channel B (ECU 3 and ECU 4).

The criterion value of the given ECU-to-channel assignment is evaluated in the

following way: If no endpoint of the hyperedge is assigned to channel B, then the set of signals it represents is transmitted only in channel A and their payload is added to the payload of A (denoted as P_A). On the other hand, if none of the endpoints are in channel A then the set of signals is only transmitted in channel B and their payload is added to the payload of B (P_B). If the hyperedge has endpoints from A and at least one from B then the set of signals must be transmitted in both channels and traverse the gateway. Their payload must be added to the payload of both channels (P_A and P_B) and the payload of the gateway (P_G). After applying the modifiers, the objective is to minimize

$$\max(\beta P_A, P_B) + \alpha \cdot P_G \tag{2.1}$$

When $\alpha = 1/\sum_{i \in S} c_i$ then the criterion ensures that among the solutions with the same channel payload, the one with a lower gateway throughput is chosen.

The well known two-partition optimization problem [43] (deciding whether a multiset of positive integers can be partitioned into two subsets such that the sum of the numbers in both subsets is the same) can be reduced to the ECU-to-channel assignment subproblem in polynomial time as follows: Each item from the multiset of the positive integers is modeled by one ECU connected to a self-loop. The weight of loop w_e is equal to the value of the item. Then the two-partition problem can be solved by an algorithm designed for the ECU-to-channel assignment. Thus, the ECU-to-channel assignment must be at least NP-hard because the two-partition optimization problem belongs to the NP-hard [43] class.

ILP model of the ECU-to-channel assignment

The exact solution can be obtained by MILP formulation. The MILP model is presented in Fig. 2.5 where x_i is a binary variable determining which channel ECU i is connected to. If $x_i = 1$ then ECU i is connected to channel A otherwise it is connected to channel B. Variable $u_{e,A} = 1$ says that hyperedge e from the set of hyperedges E is connected to at least one ECU from N that is connected to channel A (e.g. in Fig. 2.4, $u_{e_4,A} = 1$ because ECU 5 is connected to channel A). Similarly, the value of $u_{e,B} = 1$ means that the hyperedge e is connected to at least one ECU from N that is connected to channel B (e.g. $u_{e_3,B} = 1$ and $u_{e_4,B} = 0$ in Fig. 2.4). In other words, the set of signals represented by hyperedge e has to appear in the schedule for channel B. Variable z is a slack variable that helps to substitute the $max(\beta P_A, P_B)$ statement. Note that MILP formulation only contains $|N|$ integer/binary variables. The rest of the variables can be continuous because the minimization criterion ensures its integer value at any resulting solution. N_e represents the set of endpoints of the hyperedge e which are from N (ECU 3, 4, and 5) and w_e is the payload of the hyperedge.

Equations (1), (2), (3) correspond to the problem criterion. Equations (4), (5) and (6) calculate the values of P_A, P_B and P_G. Variables x_i and $u_{e,A}$ are interrelated by (7). Variables x_i and $u_{e,B}$ are interrelated by (8). For example, in Fig. 2.4, $u_{e_4,B} = 1$ because ECU 4 is one of the endpoints of e_4 and it is connected to channel B ($x_4 = 0$). Therefore, according to (8), $-0 - u_{e_4,B} \leq -1$ and after simplification $u_{e_4,B} \geq 1$. The problem is symmetric because if all ECUs are swapped to the second channel, the resulting criterion value will be the same. This symmetry

$$\min \; z + \alpha \cdot P_G \tag{1}$$

$$\text{s.t.} \quad \beta P_A \leq z \tag{2}$$

$$P_B \leq z \tag{3}$$

$$P_A + P_B - \sum_{e \in E} w_e = P_G \tag{4}$$

$$\sum_{e \in E} w_e \cdot u_{e,A} = P_A \tag{5}$$

$$\sum_{e \in E} w_e \cdot u_{e,B} = P_B \tag{6}$$

$$x_i - u_{e,A} \leq 0 \qquad \forall \{e, i | i \in N_e\} \tag{7}$$

$$x_i + u_{e,B} \geq 1 \qquad \forall \{e, i | i \in N_e\} \tag{8}$$

$$x_0 = 1 \tag{9}$$

$$\text{where:} \quad x_i \in \{0, 1\} \qquad \forall i \in N \tag{10}$$

$$u_{e,A},\, u_{e,B} \in < 0, 1 > \qquad \forall e \in E \tag{11}$$

$$P_A, P_B, P_G, z \in \mathbb{R}^+ \tag{12}$$

Figure 2.5: The MILP formulation for the ECU-to-channel assignment problem

is partially broken by Equation (9) which forces the first ECU to be connected to channel A.

The model is efficient because even if the number of variables is large (e.g. many signals in the problem) there are only a few decision variables (equal to the number of ECUs in N) and $|N| \ll |S|$ in real cases. The number of decision variables cannot be reduced as follows from the proof of NP-hardness. However, the maximal number of constraints in this problem is $6 + 2\sum_{e \in E} |N_e|$.

ECU-to-channel assignment heuristic (CAH)

Even though the MILP model scales well with respect to the number of signals, it may take a very long time to find a solution if the number of ECUs in N gets bigger or if the sets of signal endpoints are wide-ranging (which causes a large cardinality of E). It is beneficial to use a heuristic approach which finds a good solution in a reasonable time in such a case. It is even more important if CAH is a subproblem whose optimal solution does not guarantee the optimal solution of the whole problem. Thus, it is enough to have a good solution that can be obtained quickly.

The problem is similar to many problems from other areas (clustering [12,13,24], partitioning [43] or scheduling on two parallel identical resources [1]). We examined these problems and proposed our solution outlined in Algorithm 2.2. The algorithm is structured as a 3-stage local search. It uses a restart method which starts a new search from a random position of the search space to escape from a local optimum.

At the beginning of the loop, the ECUs from N are randomly ordered to the list *NL*. Then the GreedyAssignment function based on the idea of the List Scheduling algorithm [1] constructs an initial ECU-to-channel assignment solution and stores it

Algorithm 2.2 Heuristic algorithm for the ECU-to-channel assignment problem

Input: N, E
Output: Best found ECU-to-channel assignment

```
ChannelAssignment Asg, bestAsg
ECUList NL
bool changed
for triesCount do
    NL ← Randomize(N)
    Asg ← GreedyAssignment(NL, E)
    changed ← true
    while changed do
        [changed, Asg] ← ExchangeAlg(Asg)
    end
    bestAsg ← getBetter(Asg, bestAsg)
end
x ← 2-OptAlg(bestAsg)
return bestAsg
```

into *Asg*. It takes the ECUs from *NL* one by one. First, it tries to connect the ECU to channel A and evaluates the objective function for a partial assignment. Then it does the same with channel B. The ECU is assigned to the channel for which the assignment has a lower criterion value and the GreedyAssignment function continues with the next ECU from *NL*.

The initial assignment is repeatedly improved by the ExchangeAlg. The method tries to find an ECU whose move from the original channel to the second one would improve the criterion value. If such an exchange is found, the assigned channel for the given ECU is changed too. This process is repeated until there is no improvement in the assignment *Asg*. The newly obtained solution is compared to the best found assignment *bestAsg*. If it is better than *bestAsg*, the new solution is stored.

The 2-Opt [54] like algorithm tries to improve the best found assignment *bestAsg* at the end. It takes all pairs of ECUs $\langle v_i, v_j \rangle$ where v_i is an ECU that is connected to channel A and v_j is an ECU connected to channel B. Then it swaps the channels for both ECUs v_i and v_j. If the resulting criterion value is lower than the original one, the channel for v_i is set to B and for v_j to A. All the criterion evaluations are performed in the delta evaluation manner. It ensures that the computation time is not wasted on the recalculation of the whole objective function but only on the calculation of the differences to the original value.

2.3.2 Channel scheduling heuristic

The input of the channel scheduling consists of the set of signals S and the ECU-to-channel assignment *Asg*. The FlexRay static segment scheduling methods for single channel were already presented in a number of papers as described in Section 2.1.3. The key idea used here comes from [39] where the similarity of the static segment scheduling and two-dimensional bin packing problem was shown. An

algorithm outlined in Algorithm 2.3 is used in this chapter.

Algorithm 2.3 Heuristic algorithm for the channel scheduling problem

Input: S, Asg
Output: SA, SB
$SL \leftarrow$ Sort(S)
$i \leftarrow 1$
initialize SAB
while $f_{SL_i} = 1$ **do**
  PlaceToSchedule(SAB, SL_i)
  $i \leftarrow i + 1$
end
$SA \leftarrow SB \leftarrow SAB$
for i **to** $|SL|$ **do**
  $CH \leftarrow$DetermineChannel(SL_i, Asg)
  PlaceToSchedule(CH, SL_i)
end
ReorderSlots(SA, SB)

First, set S is ordered by the Sort function and stored into the ordered signal list SL. The Sort function orders signals by a stable sorting algorithm according to their decreasing payload, increasing gap between release date and deadline and increasing period. This ordering was shown to return near-optimal solutions in [19]. SL is ordered such that the fault-tolerant signals are at the beginning of the list.

Then the algorithm takes the signals one by one from the signal list SL. The schedule SAB (common to both channels) consisting only of fault-tolerant signals is created by placing the signals to the schedule by the PlaceToSchedule function in the first step.

The PlaceToSchedule function is a slightly modified signal-to-schedule placing function described in [19]. The function uses the first-fit based policy for placing signals to the schedule. It tries to find the first feasible position in the schedule (cycleID, slotID, offset in the frame) where the first signal occurrence could be placed. When this place is found, it is checked if all the other occurrences can be inserted into the schedule. If so, the signal is placed into the position. Otherwise, the algorithm looks for a new position. If no position can be found, it allocates a new slot and places the signal there.

Schedule SAB is distributed to the schedule of channel A (SA) and channel B (SB). It guarantees that the fault-tolerant signals will be transmitted to both channels at once.

Non-fault-tolerant signals are scheduled in the second step. The DetermineChannel function determines the channel CH to which the signal should be placed (it can be channel A, B or both) for each signal from SL as follows: If all receivers and the transmitter are connected to the same channel, it returns that particular channel. If the signal must be transmitted in both channels then both channels are returned. The last option occurs when all receivers and the transmitter are connected to both channels (e.g. if there would be a hyperedge between ECU1 and ECU2 in Fig. 2.4).

Then the schedule of the channel in which the signal will be transmitted is chosen according to the current volume of the payload in the channels. The one with a lower volume is returned.

The signal SL_i is placed into the determined schedule(s) by the PlaceToSchedule function afterward. Eventually, the signal image is placed to the second channel in the same manner, but the transmitter is N^{GW}. It is necessary to satisfy the precedence relations ($y_i \leq y'_i$) for these signals. However, this can be done by the modification of the release date of signal image according to the cycleID, in which the signal is transmitted.

The described algorithm does not ensure that the schedules are feasible because the transmission of the signal image s'_i can precede the transmission of the signal s_i in the same cycle (it means that $t'_i \leq t_i$). This problem is solved by the ReorderSlots function. The function reorders the slots from schedule (SA, SB) to satisfy the constraint $t_i < t'_i$. The signal images can be transmitted only by N^{GW}. Thus, the ReorderSlots function takes each gateway slot N_j^{GW} and finds the slot, in which the latest original signal is transmitted. Let us denote that slot by Q. Then it postpones slot N_j^{GW} after slot Q. Note that if slot N_j^{GW} is in the schedule for channel A then slot Q is in the schedule for channel B because only the signal images of signals from channel B can be placed in N_j^{GW}.

At the end, the schedules for both channels are obtained.

2.4 Experimental results

The proposed algorithm was coded in C++ and tested on a PC with Intel ® Core™ i7 CPU (3 GHz) and 8 GB RAM memory. A Gurobi Optimizer 6.5 was used for solving ILP formulation. The experiments were performed on a few different benchmark sets. The first one is the RealCase instance. This instance reflects the behavior of the algorithm on the set of signals from a real car of our industrial partner. It consists of 24 ECUs and 5043 signals. The five busiest ECUs transmit more than 3500 signals altogether. Almost 65% of the signals have a period equal to 40 ms. The longest signals have a payload of 4 bytes, and each signal has two receivers at most. This instance was analyzed, and its probability model was created. A new synthesized benchmark set (Synth) of 100 instances was generated according to the probability model. The rest of the benchmark sets are based on the Society of Automotive Engineers (SAE) instances generated by Netcarbench [10] and extended to include information about the signal receivers. These sets are denoted as SAE_1 ... SAE_7 and contain 100 instances each. The instances consist of more than 5000 signals that are spread to more than 22 ECUs.

The SAE benchmark sets differ from each other in the distribution of the number of signal receivers. On one hand, there are about 75% of signals with only one receiver in SAE_1. On the other hand, there are only 5% of signals with only one receiver and more than 75% of signals are received by four or more ECUs in SAE_7. The instances contain no fault-tolerant signals because the differences in the evaluations are most significant in this configuration.

The comparison of the ECU-to-channel assignment algorithms is presented in Table 2.1. The captions of the benchmark sets, which are situated in the rows,

Table 2.1: Comparison of the ECU-to-channel assignment algorithms

Set	ILP	CAH	CAH_{gap}	GA	GA_{gap}
RealCase	174843	174843	0.00‰	174843	0.00‰
Synth	314748	314844	0.30‰	316405	5.24‰
SAE_1	241516	241524	0.03‰	247237	23.14‰
SAE_2	259586	259608	0.08‰	263469	14.74‰
SAE_3	275871	275932	0.22‰	279408	12.66‰
SAE_4	300177	300300	0.41‰	302439	7.48‰
SAE_5	316842	316976	0.42‰	318516	5.26‰
SAE_6	326049	326182	0.41‰	327591	4.71‰
SAE_7	343301	343387	0.25‰	344279	2.84‰
Average	297108	297191	0.27‰	299762	9.5‰

Table 2.2: The number of the allocated slots of the static segment for individual algorithms

Set	LBSC	ISSS1	$ISSS1_{GW}$	ISSS	$ISSS_{GW}$
RealCase	210.0	121.0	13.0	121.0	13.0
Synth	219.6	160.3	40.9	158.4	40.4
SAE_1	191.4	126.1	26.1	125.6	25.9
SAE_2	191.2	134.7	32.3	133.7	31.9
SAE_3	191.8	142.0	37.5	141.0	37.0
SAE_4	191.2	152.3	46.0	151.0	45.5
SAE_5	190.9	159.7	51.7	158.4	51.4
SAE_6	191.3	164.1	54.4	162.9	53.9
SAE_7	191.2	172.1	59.9	171.1	59.4
Average	194.8	151.4	43.6	150.2	43.1

are in the first column. The second column presents the criterion value of the optimal solution of the ECU-to-channel assignment obtained by the ILP. The value is calculated according to Equation 2.1 from Sec. 2.3.1 where the α coefficient is set to $1/\sum_{i \in S} c_i$ and $\beta = 1$. The criterion value of the ECU-to-Channel Assignment Heuristic (CAH) is in the third column. The *triesCount* is set to 10 000 for CAH. Column CAH_{gap} presents the average optimality gap between the ILP and heuristic solution. It is equal to 0 ‰ for RealCase because there was just one instance and CAH found the optimal ECU-to-channel assignment for it. The fifth column presents the criterion value obtained by a binary genetic algorithm (GA). The size of the population is set to 100. Each individual is represented by an assignment vector of length $|N|$ where each binary value determines if the given ECU is assigned to channel A or channel B. The GA is stopped after 100 generations or if the number of generations without an improvement reaches 20. The last column presents the optimality gap for the results of the GA with respect to the ILP.

Table 2.2 contains the resulting number of allocated slots given by Algorithm 2.1. The feasibility of solutions was check by the inbuilt validator of Vector FIBEX Explorer. The lower bound for the single channel scheduling (LBSC) is in the second column. The computation of the lower bound is derived from the lower

Table 2.3: The computation times of different algorithm variants in ms

Set	ILP	CAH	GA	ISSS1	ISSS
RealCase	1163	1121	826	313	996
Synth	21841	2696	3094	571	4028
SAE_1	16942	1377	2193	370	1491
SAE_2	89943	2354	3401	502	2753
SAE_3	232520	3027	4260	665	4902
SAE_4	722658	4676	5906	865	7063
SAE_5	1521635	6213	7276	1120	11401
SAE_6	1964601	7364	8177	1275	12762
SAE_7	3062509	9316	9387	1596	15212
Average	810384	4238	4947	870	7443

bound for the 2D bin packing which is evaluated for each ECU separately. The LBSC is then the sum of the rounded up lower bounds of the ECUs. The third column (ISSS1) contains the average number of slots allocated by the Iterative static segment scheduling heuristic after the first iteration of the algorithm, and the fourth column ($ISSS1_{GW}$) contains the number of slots used by the gateway ECU. The same values for the best found schedule by the algorithm are in the fifth and sixth column. The Iterative Static Segment Scheduling heuristic (ISSS) uses the CAH for ECU-to-channel assignment with *triesCount* equal to 1 000.

The average execution times, in milliseconds, of each individual part of the algorithm are presented in Table 2.3. It is organized in a similar way to Table 2.1.

A new set of instances based on SAE_1 set was created such that each subset of 100 instances has a different percentage of fault-tolerant signals and common ECUs. The percentage varies from 0% to 100% with the step of 5%. Fig 2.6 presents the dependence of the number of allocated slots on the percentage of common ECUs and the percentage of fault-tolerant signals from these ECUs. It can be observed that the results are strongly affected by the percentage of fault-tolerant signals for the case with a big percentage of common ECUs. Theoretically, the bandwidth used by a schedule with no fault-tolerant signals is equal to 50% of the bandwidth used by the schedule where all the signals are fault-tolerant if all the ECUs are common.

From Table 2.1, it can be observed that CAH finds a near-optimal solution (about 0.2‰ gap in average). It appears that, for comparable computation times, the results obtained by CAH are closer to the optimal value in comparison to GA. Furthermore, CAH returns an optimal solution in 633 out of 801 cases, and its result is obtained significantly faster with respect to ILP.

According to the ISSS and LBSC results, the usage of the independent channels with gateway can save about 10% to 45% of the bandwidth. It strongly relies on the number of signal receivers and their diversity. It is possible to save about 45% of the slots if the diversity is small (e.g. in the RealCase). At the other extreme, SAE_7 contains signals with a lot of receivers per signal, and the diversity of the signal receivers is large. Therefore, there is a saving of just about 10%. The use of the iterative algorithm encapsulating the ECU-to-channel assignment and the channel scheduling helps to save about one static slot on average. The interesting

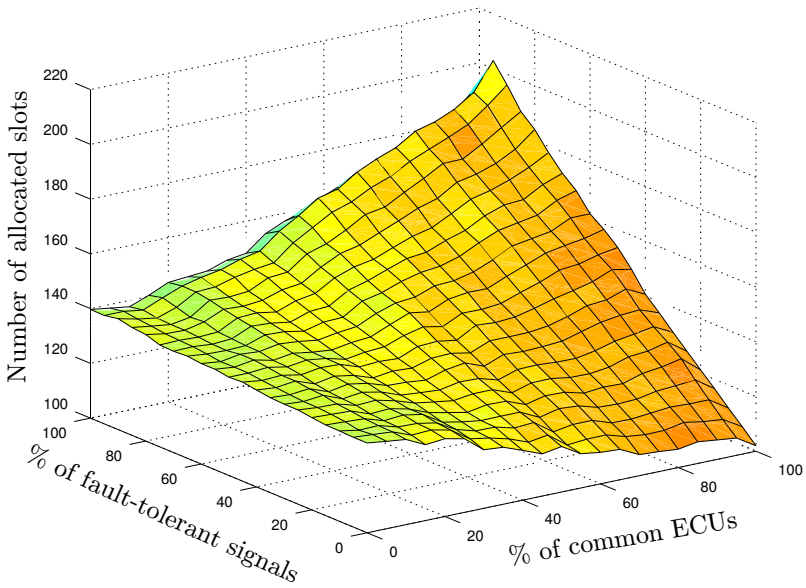

Figure 2.6: The dependence of the number of allocated slots on the percentage of common ECUs and fault-tolerant signals

observation is that it does not only help to balance the schedule but it also slightly decreases the number of slots transmitted by the gateway ECU.

It is possible to observe from Table 2.3 that the execution time of the ILP increases significantly with the increasing number of signal receivers even for a similar number of signals and ECUs. It is caused by the signal aggregation which is not able to reduce the number of hyperedges in these instances.

2.5 Conclusion

Even if the Automotive Ethernet is being pursued to replace the currently used buses in vehicles, it will not be capable of handling high-critical applications in a reliable way in near future. Thus, the efficient FlexRay communication scheduling is a crucial problem for applications as x-by-wire or chassis control systems. The solution that aims to save the bandwidth by utilizing the benefits of the independent channels and efficient scheduling was described in the chapter.

We developed the heuristic algorithm which decomposes the complex problem to two subproblems, the ECU-to-channel assignment subproblem and the channel scheduling subproblem, and iteratively solves the subproblems with modified values of the parameters. The proof of the NP-hardness and the efficient ILP model were provided for the ECU-to-channel assignment subproblem. Furthermore, the polynomial time local search algorithm, which returns a near-optimal solution, was designed to deal with the computational complexity of the exact algorithm. The channel scheduling subproblem for placing both fault-tolerant as well as non-fault-tolerant signals was solved heuristically by the first fit policy based algorithm.

Obtained schedules are feasible according to the FlexRay specification.

The evaluation of the algorithm on the synthesized and real problem instances showed that utilizing the independent channels can save around 30% of the single channel bandwidth depending on the diversity of the signal recipients. The problem appeared to be sensitive to the percentage of fault-tolerant signals in cases with a high percentage of common ECUs. This feature predetermines our approach mainly for applications where the percentage of fault-tolerant signals is not too high and the number of signal receivers is relatively small.

Chapter

3 Multi-variant scheduling of critical time-triggered communication in incremental development process: Application to FlexRay

The portfolio of models offered by car manufacturing groups often includes many variants (i.e., different car models and their versions). With such diversity in car models, variant management becomes a formidable task. Thus, there is an effort to keep the variants as close as possible. This simple requirement forms a big challenge in the area of communication protocols. When several vehicle variants use the same signal, it is often required to simultaneously schedule such a signal in all vehicle variants. Furthermore, new vehicle variants are designed incrementally in such a way as to maintain backward compatibility with the older vehicles. Backward compatibility of time-triggered schedules reduces expenses relating to testing and fine-tuning of the components that interact with physical environment (e.g., electromagnetic compatibility issues). As this requirement provides for using the same platform, it simplifies signal traceability and diagnostics, across different vehicle variants, besides simplifying the reuse of components and tools.

This chapter proposes an efficient and robust heuristic algorithm, which creates the schedules for internal communication of new vehicle variants. The algorithm provides for variant management by ensuring compatibility among the new variants, besides preserving backward compatibility with the preceding vehicle variants. The proposed method can save about 20% of the bandwidth with respect to the schedule common to all variants. Based on the results of the proposed algorithm, the impact of maintaining compatibility among new variants and of preserving backward compatibility with the preceding variants on the scheduling procedure is examined and discussed. Thanks to the execution time of the algorithm, which is less than one second, the network parameters like the frame length and cycle duration are explored to find their best choice concerning the schedule feasibility. Finally, the algorithm is tested on benchmark sets and the concept proved on the FlexRay powered hardware system.

3.1 Introduction

The cars currently being produced by automotive industry contain a lot of electronic control units (ECUs), which are becoming progressively more important in the upcoming vehicle models, where x-by-wire systems should be able to replace mechanic and hydraulic control systems. This approach has been already used by,

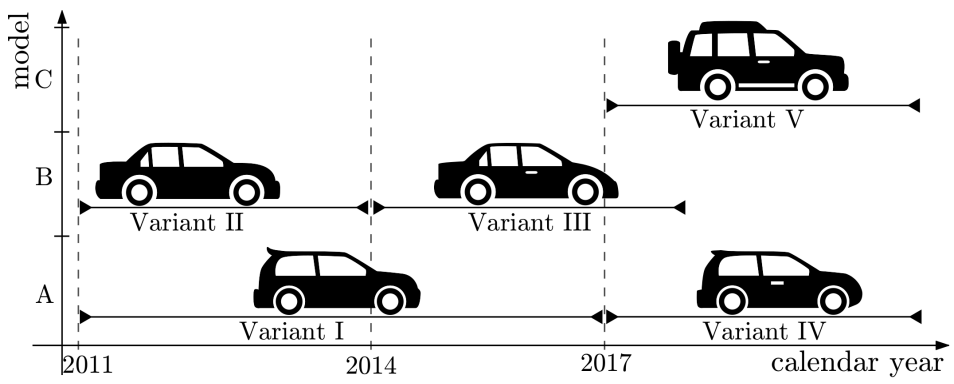

Figure 3.1: Example of product lifetimes

for example, Nissan Infinity Q50 called as Direct Adaptive Steering technology [40]. The spectrum of the electronic systems used, however, varies from one model to the other.

3.1.1 Motivation

Nowadays, the car manufacturers have to manage a huge number of model variants. For example, Volkswagen group proclaimed in [66] that their product portfolio consists of 340 model variants already. Handling such a large number of variants is indeed challenging for the car designers. The basic approach adopted in dealing with such situations is by building most vehicle models on a common technological platform. The vehicle models, such as the Audi A3, SEAT Leon, Volkswagen Golf and Škoda Octavia, for example, share a modular construction of the MQB (Modularer QuerBaukasten) platform [11]. Moreover, these vehicle models also have many versions (e.g., a configuration with an adaptive LED frontlight system or with common halogen lamps, etc.). Thus, an efficient variant management can be a significant economical and competitive factor [49].

Having the internal vehicle communication as similar as possible for all the vehicle variants (referred to as *variants* hereafter) is desirable to simplify the reuse of components and decrease the development costs spent, for example, on fine tuning of electromagnetic compatibility related parameters. Creating just one communication schedule for all the signals of different variants would be ideal from this perspective. However, such a schedule results in low utilization of the bus, because each variant uses only a subset of the signals. It is important to have a bandwidth-efficient solution, because the demand for communication bandwidth has been increasing significantly, while transmission of messages from a camera or a lidar becomes a part of safety-related systems in modern vehicles. These bandwidth consumming messages will be evan more critical for a safety reinsurance in autonomous driving systems. Therefore, some systematic solution will have to be found to utilize the bandwidth efficiently. In the present case, the design practice, derived from the designer requirements, has been followed, wherein the same signals are placed at the

same positions in all the schedules in which they participate. This method facilitates the trade-off between compatibility of the schedules among the variants and their bandwidth efficiency. Consequently, each vehicle variant will have a schedule, which differs only in the positions of specific signals. Therefore, the objective of this study is to find a multivariant schedule, which includes individual schedules for all variants.

Furthermore, designing a car is an iterative process; while the previous variant is in production, the new one is in the design stage. An example of product lifetimes is depicted in Fig. 3.1. While designing a new variant, the previous variants cannot be changed, but the new variant should maintain backward compatibility with the previous variants, to the extent possible. This is called the *incremental design process* during which the new variant is not built from scratch, but the variant is based on its predecessors. This implies that, if a new vehicle variant is being developed, its schedule should be inherited from the original variant.

It is also necessary to proactively enhance the extensibility of the schedule, so that it can create compact schedules in all the development iterations, because the schedule will be probably considered as the original schedule (the schedule on which the schedule for the new variant is based) for successive iterations. This type of incremental problem is even more challenging, because, for example, the inheritance from more than one predecessor can entail conflicts that need to be resolved with the least number of disruptions for ensuring backward compatibility.

With such a multischedule, it is easier and cheaper to develop diagnostic tools, because one tool can be used for many variants. Also, it simplifies the configuration of electronic control units (ECUs), typically supplied by third parties, because one bus configuration of the ECU may fit several variants. This eliminates many mistakes, (e.g., relating to time dependent electromagnetic interference), and thus reduces verification and certification expenses. Additionally, it reduces the likelihood of failure during the verification process, and hence likelihood of additional redesign costs and, consequently, postponing the release date of the product [48].

In this chapter, the focus is on the design and implementation of the algorithm for solving the above described multi-variant and incremental scheduling problems. For verification and demonstration of the algorithm, the FlexRay static segment has been chosen as the protocol for time-triggered communication. The FlexRay standard has been designed to handle the safety and criticality-related requirements on the complex interconnected electronic system. A static segment of the FlexRay protocol, with time division multiple access, can be used for time-critical signals, which need to fulfill real-time constraints as release date or deadline. The signals are to be transmitted to the bus at exact time instants, as determined by a schedule, which must be known in advance.

3.1.2 Related works

Product variant management is a problem faced by many companies, because they have to fulfill the individual requirements of their customers. Bley and Zenger [7] investigated this problem in the planning process of an assembly, whose final product consists of many parts. According to Wallis et al. [67], this problem is even more relevant to digital factories, wherein, nowadays, digital manufacturing data provides

the information necessary for automatic planning of production. A similar problem has been tackled in software development process. Variants of the software product and their source codes need to be managed carefully [69, 70] otherwise the product becomes uncontrollable with smelly code [23]. Sagstetter et al. [49] observe that, by rapidly increasing number of vehicle model variants, variant management becomes an important and challenging problem for the fast evolving automotive industry. They have shown that vehicle variant management is strongly linked to the creation of time-triggered communication schedules.

A significant effort has gone into developing a methodology for finding a reliable, time-deterministic and bandwidth-efficient communication schedule for suitable in-vehicle networks, such as Ethernet, FlexRay or TTP.

For TTEthernet and Automotive Ethernet, Steiner et al. provide the whole scheduling and time analysis framework [15, 57, 58, 60]. They used SMT Solver, Tabu Search and Network calculus to create a schedule of time-triggered traffic and evaluate the schedule from the event-triggered communication point of view. More detailed description of works related to Automotive Ethernet is present in Chapter 4

Several papers that focus on the FlexRay protocol, and particularly the static segment scheduling problem, were published during the last ten years. The mathematical basics required for scheduling time-triggered and event-triggered communication were laid down by Schmidt and Schmidt [50, 51]. Their scheduling method was based on ILP formulations for signal-to-frame packing and frame scheduling. In the static segment scheduling area, Lukasiewycz et al. [39] are the pioneers in introducing the method for transformation of the basic static segment scheduling problem, without time constraints, into a two-dimensional bin packing problem. Their objective is primarily to minimize the number of the allocated slots and, secondly, to obtain such a schedule that can accommodate further signals with no need for allocation of new slots. They also explain how their algorithm behaves in the case of incremental scheduling, where no conflicts can occur. Hanzalek et al. [26] propose the static segment scheduling problem with real-time constraints. They present a two-stage scheduling algorithm; in the first stage, the signals are packed into the frames and in the second, the schedule is created by a frame scheduling algorithm. The reliability of the broadcast FlexRay communication was studied by Souto et al. [18].

The methods for an extensible TTP protocol scheduling, based on the original schedule, are described by Pop et al. [44]. They first find the solution that satisfies the hard real-time constraints, and then they try to improve the availability of the resource for further use by the iterative algorithm.

Of late, some scientists have been focusing on cooperation of the deterministic buses in automotive industry. The key component of the reliable system, is the gateway that interconnects its constituent heterogeneous buses. After studying the problem of time synchronization between FlexRay and Ethernet [37], Jeon et al. proposed a framework for reliable gateway development [33].

A proposal for Multi-variant Scheduling was first presented in [19], wherein schedules for more variants were created, all at once. This was later followed up by Sagstetter et al. [49] who have iteratively constructed a multi-schedule, in which the signals common to all the variants are scheduled in the first iteration, the shared

signals in the intermediate iterations, and the signals specific to just one variant in the last iteration. To the best of the authors' knowledge, all the published methods for scheduling the time-triggered communication have been built on greenfield, without considering previous/original schedules.

The 2D bin packing problem is closely related to the time-triggered scheduling, as observed by Lukasiewycz et al. [39]. The main problems of incremental scheduling for bin packing was investigated by Gutin et al, who presented the lower bound for the asymptotic competitive ratio of any algorithm [25]. Later, Ivković and Lloyd [30] described the algorithm and its $\frac{5}{4}$ competitive ratio for the fully dynamic case of bin packing problem, with Insert and Delete operations. They classified the items into groups, according to their size.

3.1.3 Outline of the chapter

The rest of the chapter is organized as follows: Section 3.2 describes the incremental static segment scheduling problem, encompassing the real-time constraints and providing multi-variant scheduling scheme, together with a brief example; Section 3.3 introduces the data structures which support the efficiency of the proposed algorithm and then proposes an efficient heuristic algorithm for incremental scheduling; Section 3.4 presents the signal set and the experiments carried out on the extensive benchmark set; finally, Section 3.5 concludes the chapter.

3.2 Problem statement

3.2.1 Periodic scheduling with real-time constraints

The scheduling problem addressed in this chapter is as follows: Let S be a set of all *signals* that must be exchanged. Each signal $s_i \in S$ has the following parameters:

- p_i - period
- c_i - payload length
- n_i - identifier of the transmitting ECU
- r_i - release date
- d_i - deadline

According to the AUTOSAR Specification, the period p_i is a multiple of power of two (i.e., $p_i \in \{L \cdot 2^l \mid l = 0 \dots 6\}$), where L is the duration of one communication cycle. The payload c_i of a signal must be in the range of 0 to 254 bytes. The signal must be transmitted by the ECU as a whole, without being fragmented. The unique identifier n_i determines which ECU transmits signal s_i. Real-time constraints are represented by the release date r_i and deadline d_i. Both the parameters are considered to be relative to the beginning of the schedule and it is supposed that $d_i \leq p_i$. In order to simplify the problem, release date r_i and deadline d_i are considered to have been rounded to the length of the cycle (as in [26]). This simplification is adequate, because the precise specification of the release dates and

deadlines will have influence only if these values fall in the static segment. However, if they fall in the dynamic segment, they are rounded to the length of the cycle anyway. This rounding simplifies the scenario, because the position of a signal within the static segment of a particular cycle is insignificant, compared to that of the signal within the hyperperiod. Rounding of the release dates and deadlines allows the scheduling of each ECU separately (because the position of the slots of the given ECU is not important from the viewpoint of real-time constraints), which reduces the combinatorial complexity of the problem.

The FlexRay network configuration consists of many parameters. The following parameters, which are assumed to have been chosen by network designers, are not influenced by the optimization algorithm:

- L - duration of the communication cycle,
- W - maximal frame payload length (duration of the slot).

We assume that the number of slots in the static segment of the communication cycle does not exceed slots threshold (i.e., the maximal number of slots that fits one communication cycle), and, thus, is sufficient to accomodate the generated schedule. Section 3.3.5 discusses how to deal with the case when the number of the allocated slots exceeds the slots threshold.

The aim of the scheduling problem is to find a schedule - an assignment $s_i \rightarrow [y_i, t_i, o_i]$, where

- y_iIdentifier of the communication cycle represents the identifier of the communication cycle (cycleID) for the first *signal occurrence* (instance of the signal in the hyperperiod),
- t_i denotes the identifier of the slot (*slotID*),
- o_i is the offset in the frame (*offset*) in which the first occurrence of the signal s_i will be transmitted.

The signals are assumed to be strictly periodic (no jitter in period p_i is allowed). Thus, all the other signal occurrences are scheduled at the same slotID and offset. The cycleID of j-th signal occurrence is calculated from the cycleID of the first occurrence and its period p_i as $y_i + (j-1)p_i$. The goal is to find such an assignment, in which $\max_{i \in S}\ t_i$ is minimal.

3.2.2 Multi-variant scheduling

Considering the multi-variant scheduling, the following holds:

- Each signal s_i can be used in one or more variants.
- **Sharing constraint**: If two or more variants use signal s_i, the signal must be placed in the same position (cycle, slot, even offset in the frame) of these schedules.
- The slots assigned to some ECU are assigned to this ECU in all the variants in which the ECU is used.

These are the reasons why it is not possible to create schedules for all variants independently. To identify which variant uses which signals, the binary matrix V is introduced as follows:

$$V_{i,j} = \begin{cases} 1, & \text{if variant } j \text{ contains signal } s_i. \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

The resulting schedule must fulfill all the described constraints. Moreover, no two signals are allowed to overlap in any variant.

3.2.3 Incremental scheduling

For incremental multi-variant scheduling problem, original assignment $s_i \rightarrow [\tilde{y_i}, \tilde{t_i}, \tilde{o_i}]$ is defined for the subset of signals $s_i \in \tilde{S}$ where $\tilde{S} \subset S$. The assignment $s_i \rightarrow [\tilde{y_i}, \tilde{t_i}, \tilde{o_i}] | \forall s_i \in \tilde{S}$ is called the *original schedule* in this chapter.

The aim of incremental scheduling is to find the assignment for all the signals from S, which fulfills all the constraints of the multi-variant scheduling, minimizes $\max t_i$ and where the number of changes compared to the original schedule is minimal.

It is to be noted that not only new signals, but even new variants are introduced in incremental multi-variant scheduling. It follows the real case, when a new vehicle variant is proposed. This is the reason why even signals from the original schedule could cause a violation of the constraints because a newly introduced variant can use two signals that overlap in the original schedule (because they were not used in any variant together so far).

Example 1: A simple example of incremental scheduling

A simple example is introduced for a better understanding of the given problem statement. Here, the communication cycle duration L is set to 5 ms and the frame payload W to 16 bits. The original schedules for Variant I and Variant II are depicted in the lower part of Fig. 3.2. In this figure, the individual communication cycles are placed, one below the other, in vertical rows, in contrast to those in Fig. 2.1, where they are placed laterally, one next to the other, in the timeline. Moreover, only the static slots of the communication cycles are presented in Fig. 3.2. This visualization will be used hereafter. In total, there are eight signals, $s_1 \ldots s_8$, which are to be transmitted from three ECUs. The identifier of the ECU, assigned to a slot, is determined by the pale label above the slot. Thus, one can derive from the figure that, for example, signal s_1 is to be sent by ECU 1 in Variant I only, it has the period 5 ms and the payload 8 bits. The deadlines and release dates of all the signals are ignored in this simple case. Variant I uses signals s_1, s_2, s_3, s_6, s_7 and Variant II uses $s_2, s_3, s_4, s_5, s_6, s_8$, which means that signals s_2, s_3, s_6 are shared and must be placed in the same position in both the variant schedules (Venn diagram for the variants is depicted in the upper part of Fig. 3.2).

Now, the task is to create the new variant - Variant III, which should use all the signals of Variant I and Variant II and, furthermore, it should also accommodate new signals s_9 and s_{10}. Signal s_9 has the period of 5 ms and is transmitted by ECU 2. Signal s_{10} has the period of 10 ms and is transmitted by ECU 1. The

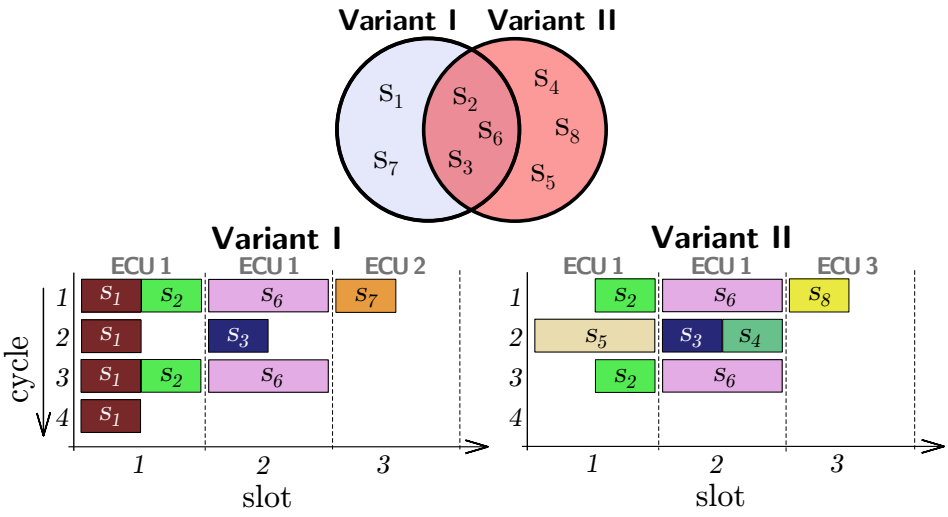

Figure 3.2: Venn diagram for Variant I and Variant II, and its original schedules for Example 1

Venn diagram for all the three variants and the resulting schedule are presented in Fig. 3.3.

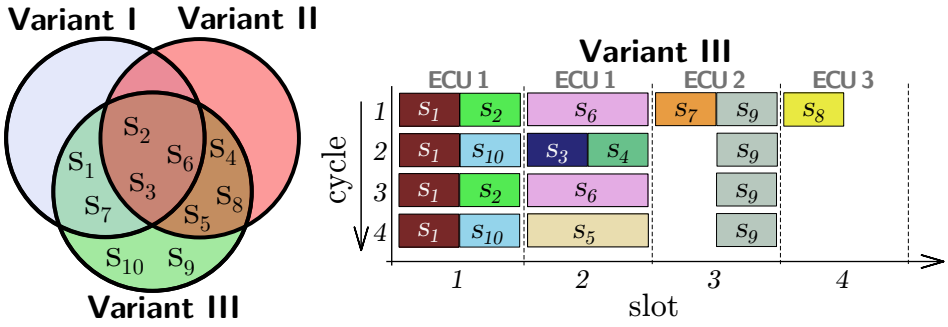

Figure 3.3: Venn diagram for all the three variants, and a feasible schedule for the new Variant III

All the common signals of the new variant should be placed in the same positions as those in Variants I and II. However, it is not always possible to satisfy this backward compatibility constraint in incremental multi-variant scheduling, and some signals, signals s_5 and s_8 in this example, must be rescheduled to prevent collisions. In the case of s_8, not only the signal had to be moved, but the entire slot has to be moved from the third slot to the fourth slot. Otherwise, both ECU 2 and ECU 3 would operate in slot 3, which is not allowed.

3.3 Algorithm

In this section, the main data structures used in the proposed algorithm are introduced first and then the components of the proposed algorithm explained.

3.3.1 Multischedule

For schedule representation, choosing the right data structure is crucial to the efficiency of algorithm. The most natural way is to have different schedules for different variants (as in Fig. 3.2), which are called here as native schedules. However, this representation renders the algorithm inefficient, because the checking of the sharing constraint and the allocation of signals in native schedules introduce significant overhead, when the common signals increase in number. It is enough if their position is known just in one schedule, because the position must be the same for all the variants.

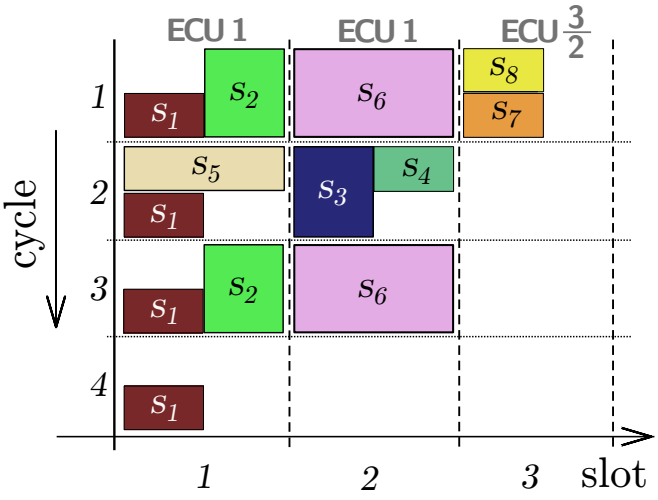

Figure 3.4: Feasible original multischedule for Example 1

Therefore, a more efficient representation is used by creating just one shared schedule, called *multischedule*, for all the variants, instead of separate native schedules for each variant. The common signals are placed once, and there is no redundancy caused by checking the constraints. Native schedules are derived from the multischedule by removing the signals that have not been used in the particular variant. In the multischedule, two or more signals may be scheduled at the same position (this situation is denoted as overlapping). Just as the native schedule consists of frames, the multischedule (MS) consists of multiframes, which are denoted as $MS_{i,j}$, where i is the cycle number and j is the slot number. The original multischedule (the multischedule derived from the original schedules) from Example 1 is presented in Fig. 3.4, where the lower half of each multiframe represents the first variant and the upper half the second variant. An example of signal overlapping can be seen in $MS_{2,1}$ where signal s_5 shares its position with that of signal s_1.

3.3.2 Mutual exclusion matrices

For placing signals in the multischedule, one needs to know the signals that may be overlapped. Overlapping can arise only when two signals are not scheduled in the same variant; otherwise, it would result in an infeasible native schedule for a variant that uses both the signals. Information relating to two given signals that can overlap is stored in a *Signal Mutual Exclusion Matrix* (*SEM*), which is a symmetric binary matrix, generated from matrix V. $SEM_{i,j}$ is equal to 1 if, and only if, signals s_i and s_j are to be scheduled together in some variant, otherwise, 0. Thus, two

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
s_1	1	1	1	0	0	1	1	0
s_2	1	1	1	1	1	1	1	1
s_3	1	1	1	1	1	1	1	1
s_4	0	1	1	1	1	1	0	1
s_5	0	1	1	1	1	1	0	1
s_6	1	1	1	1	1	1	1	1
s_7	1	1	1	0	0	1	1	0
s_8	0	1	1	1	1	1	0	1

Table 3.1: Signal mutual exclusion matrix for Example 1

signals s_i and s_j can overlap only if $SEM_{i,j} = 0$. This holds only for the pairs $\{s_1, s_4\}$; $\{s_1, s_5\}$; $\{s_1, s_8\}$; $\{s_4, s_7\}$; $\{s_5, s_7\}$ and $\{s_7; s_8\}$ in Variants I and II, from Example 1 shown in Table 3.1.

The multischedule has one extra feature in addition to the native schedule. As can be seen in Fig. 3.4, one slot in the multischedule can be occupied by more than one ECU. In the present case, signals s_7 and s_8 are scheduled in multiframe $MS_{1,3}$. However, signal s_7 is from ECU 2 and signal s_8 is from ECU 3. This results in a feasible multischedule only if the signals from these two ECUs do not appear in the same variant. The information is represented by the *ECU Mutual Exclusion Matrix* (EEM). $EEM_{i,j}$ is equal to 1 if, and only if, ECUs i and j appear in some variant together, otherwise, 0. The *EEM* matrix for Variant I and Variant II of Example 1 is shown in Table. 3.2.

	ECU 1	ECU 2	ECU 3
ECU 1	1	1	1
ECU 2	1	1	0
ECU 3	1	0	1

Table 3.2: ECU mutual exclusion matrix for Example 1

3.3.3 Conflict graph

New variants are added to the multischedule during the incremental scheduling. These new variants can cause an unavoidable violation of backward compatibility because two signals that were placed in the same position can appear in the new

variant together. To minimize the number of such violations, tracking of these signal conflicts is necessary. The *conflict graph CG* is introduced for the purpose.

Each node from the set of nodes from the conflict graph N_{CG} represents a signal that conflicts. The undirected edges E_{CG} then represents the conflict itself. Moreover, the nodes can be marked by the weighting function to express the significance of backward compatibility violation when the position of the signal, represented by the node, is changed. The used weighting function will be described in detail later in the chapter.

3.3.4 Incremental scheduling algorithm

The main idea of incremental multi-variant scheduling algorithm, depicted in Fig. 3.5, is to place a signal in the multischedule, according to a given order (described in Sec. 3.3.4). The algorithm is divided into three stages, A, B and C. In Stage A, algorithm initialization and signal sorting are performed.

In Stage B, the signals are placed in *unit multischedules* (one per-ECU). Because the entire slot is reserved for a particular ECU, and no two slots can overlap in a native schedule, a unit multischedule is made for each ECU separately. The unit multischedule decides the cycleID and the offset in the frame from which a particular signal will be sent. Moreover, for each ECU it is known as to how many slots are to be allocated by the algorithm in the final multischedule, at the end of Stage B. While Stage B is executed on per-ECU basis, the computational complexity of the algorithm is reduced following the divide-and-conquer paradigm.

During Stage C, the slots from unit multischedules are merged into the final multischedule. The following is the detailed explanation for each part of the algorithm:

Ordering of the Signal set

Order of signals is important, because the signals are placed into the multischedule, one by one. In [39], the authors have shown that the 2D bin-packing problem and the static segment scheduling problem have similar features. They propose to organize the signals in the order of their increasing period. This ordering works well when no time constraints are defined. Therefore, the proposed algorithm uses a combination of multiple orderings. The signal set is sorted in three steps (i.e. the sorting criteria are applied one after the other while a stable sorting algorithm is used), according to the decreasing payload, increasing window (gap between the release date and the deadline) and increasing period.

Sorting according to the decreasing payload and the increasing period ensures that most bandwidth-demanding signals are scheduled first; less bandwidth-demanding signals, which are more suitable for filling small gaps of the remaining bandwidth, are scheduled later. Ordering according to the increasing window ensures that the signals which are hard to schedule (their placement in the multischedule is more limited by the time constraints) are scheduled sooner. This policy has been found to obtain the most efficient solutions for non-incremental scheduling, as shown in [19], where different scheduling policies were empirically examined.

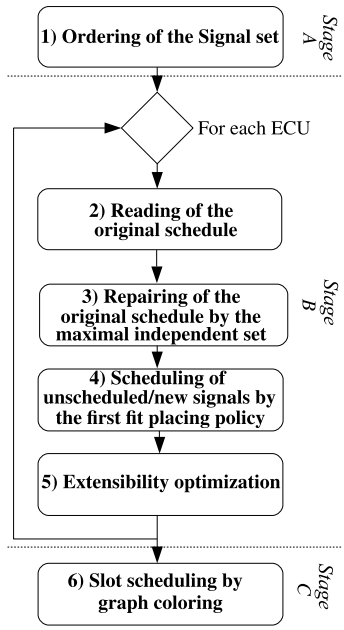

Figure 3.5: Flow chart of the algorithm

Reading of the original multischedule

At the beginning of Stage B, it is necessary to read the positions of the signals from the original multischedule MS^O and place all their occurrences at the same position in the unit multischedule MS^{ECU} of the currently scheduled ECU. If the signal is not placed in the original schedule, it is added to the set of new signals SL^N that need to be scheduled later. The pseudocode is shown in Alg. 3.1.

The currently placed signal occurrence can violate a constraint which prevents the simultaneous transmission of two signals. This happens if the newly introduced variant has two signals that were never used together in any variant, and were occupying overlapping positions in the original multischedule (e.g., signals s_1 and s_5 in Example 1). Hence, it is important to check this violation after placing each signal occurrence, by using the method FindConflictingSignals. The *SEM* matrix was used for this purpose. If two signals s_i and s_j overlap and the value of $SEM_{i,j} = 1$, then the violation occurs. If this violation arises, unordered pair(s) $\{s_i, s_j\}$ are added to the conflict graph CG as new edge in E_{CG}, where s_i is the current signal and s_j is the signal that conflicts with s_i. At the end, the CG contains all signals with a conflict.

In Example 1, while scheduling unit multischedule in cycle 2 of ECU 1 for the new Variant III, the signal s_5 conflicts with signal s_1.

Repairing of the original multischedule

After reading of the original multischedule, the violations are still included in the original unit multischedule, but they are represented by CG. The only way to

Algorithm 3.1 Reading of the original multischedule

Input : Original multischedule MS^O,
Ordered set of signals SL,
Signal Mututal Exclusion Matrix SEM
Output : Original unit multischedule MS^{ECU},
Conflict graph CG, Set of new signals SL^N

for *each signal* s_i *in* SL *transmited by the ECU* **do**
  if s_i *is in* MS^O **then**
    place signal s_i into MS^{ECU} at the same position as in MS^O
    $CG \leftarrow$FINDCONFLICTINGSIGNALS(s_i, MS^{ECU}, SEM)
  else
    $SL^N = SL^N \cup s_i$
  end
end

avoid the violation is to remove some conflicting signals from the unit multischedule. It is beneficial, as expressed by the objective, to keep as many signals (their occurrences) in the original positions as possible to have the minimal number of backward compatibility violations. This sub-problem can be expressed as the maximal independent set problem (MIS) in the conflict graph. The resulting subset of signals is denoted as S_{MIS}.

In Example 2, let a new set of original variants - Variant I to Variant III as shown in the upper part of Fig. 3.6, to be considered. In the current scheduling iteration, what is needed is the creation of new Variant IV that contains all the signals from all the original variants (Variants I to III). The graph that represents CG for ECU 1 of the given Example 2 is shown in the lower part of Fig. 3.6. This

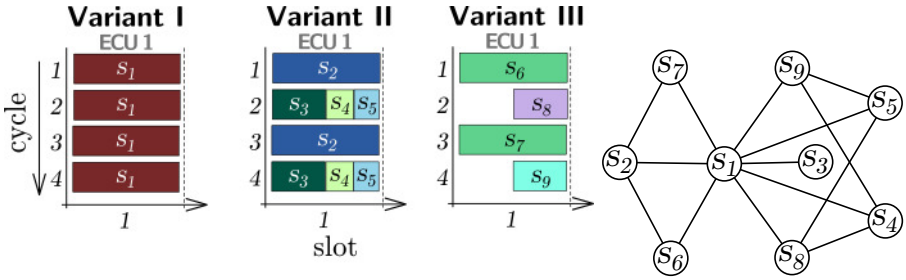

Figure 3.6: Example 2 - The conflict graph CG for scheduling Variant IV containing all the signals from variants I, II and III

example is rather simple, but it can be much more complicated in real situations.

If the subset S_{MIS} is the solution of MIS over the conflict graph, then $N_{CG} \backslash S_{MIS}$ will be the minimal subset of signals, whose removal would solve all the conflicts. In the case of Fig. 3.6, the solutions $\{s_3, s_4, s_5, s_6, s_7\} = S_{MIS}$ and $\{s_3, s_6, s_7, s_8, s_9\} = S_{MIS}$ are equivalent from the viewpoint of the number of signals. However, solution $\{s_3, s_4, s_5, s_6, s_7\} = S_{MIS}$ is assumed to be better, because signals s_4 and s_5 have

two occurrences each, against signals s_8 and s_9, which have just one occurrence. Considering this, it is preferred to remove the signals with fewer occurrences. Therefore, the algorithm uses priorities for solving this problem. The number of signals has a higher priority, compared to the number of occurrences of the signal, which is achieved by using the proper conflict graph nodes weighting function $w_i = |N_{CG}| + \frac{1}{p_i}$. According to the formula, adding one extra signal to S_{MIS} increases the objective value by at least $|N_{CG}|$. Taking into account that $p_i \geq 1$, the sum of $\frac{1}{p_i}$ over all nodes in the conflict graph cannot exceed the value $|N_{CG}|$. It ensures that any change in the number of signals used in S_{MIS} is more significant than the change in the number of signal occurrences. Thus, the weight w_i reflects the priorities. The extension of MIS to the maximal weighted independent set problem (MWIS) is needed for this purpose.

ILP model (3.2) is employed for solving MWIS thus:

$$
\begin{array}{lll}
\max\limits_{\vec{x}} & \sum\limits_{i|s_i \in N_{CG}} w_i x_i & \\
\text{subject to} & x_i + x_j \leq 1, & \forall i,j \mid \{s_i, s_j\} \in E_{CG} \\
\text{where} & w_i = |N_{CG}| + \dfrac{1}{p_i} & \forall i \in N_{CG} \qquad (2) \\
& x_i \in \{0,1\} & \forall i \in N_{CG}
\end{array}
$$

In the model, N_{CG} represents the set of nodes in conflict graph CG. Signal s_i belongs to the subset S_{MIS} if, and only if, $x_i = 1$. Signals from $N_{CG} \setminus S_{MIS}$ are, consequently, removed from the unit multischedule.

Scheduling of unscheduled/new signals

New signals and conflicting signals removed from the original multischedule are put in an ordered set, called the signal list SL^N (the list contains $\{s_1, s_5\}$ for ECU 1 in Example 1). This list is ordered as explained in Sec. 3.3.4. Algorithm 3.2 takes the signals, one by one, from the SL^N and tries to place their first occurrence in the first feasible position, in the unit multischedule, using the FindPositionForSignal method. It checks all offsets in the first frame (the frame in the first slot and in the cycle of the signal's release date) first. If it is not possible to place the signal there, the algorithm keeps repeating this exercise up to the cycle, determined by the signal's deadline. If it is still not possible to find a place in the first slot, the algorithm continues trying to find a place in the second slot, third slot and the subsequent ones. Once the feasible position for the first occurrence is found, the other occurrences are also checked for feasible placement.

If no position is available for the signal in the unit multischedule, the algorithm calls for the AllocateSlot method to allocate a new slot and places the signal into it with the cycleID equal to the release date and offset equal to 0. This procedure is repeated until all the signals from the SL^N are scheduled.

Using this one-shot constructive heuristics for scheduling instead of two-stage heuristics (e.g., the one introduced in [26]) has a significant benefit as it can schedule signals with a larger period in frames with a shorter period.

Algorithm 3.2 Scheduling of unscheduled/new signals

```
Input  : Original unit multischedule MS^ECU,
         Ordered set of unscheduled/new signals SL^N,
         Signal Mututal Exclusion Matrix SEM,
         Conflict graph CG
Output : Unit multischedule MS^ECU
for each signal s_i in SL^N do
    infeasiblePosition ← true
    while infeasiblePosition = true do
        placePosition = FindPositionForSignal(MS^ECU, s_i, SEM)
        if placePosition not found then
            break
        end
        (cycle, slot, offset) ← placePosition
        infeasiblePosition ← false
        while cycle < hyperperiod do
            if (cycle, slot, offset) is not suitable for signal then
                infeasiblePosition ← true
                break
            end
            cycle += p_i
        end
    end
    if infeasiblePosition then
        placePosition ←AllocateSlot(MS^ECU, s_i)
    end
    place signal s_i into MS^ECU at placePosition
end
```

Extensibility optimization

Now, the final number of slots that the ECU will maintain is known. However, it is needed to care about the future signals for the incremental scheduling and proactively enable their insertion. Experiments with the non-incremental problem have shown that the proposed scheduling algorithm allocates a near optimal number of slots, when all the signals are ordered according to the procedure outlined under Sec. 3.3.4. This is not true in the case of incremental scheduling, because, for example, a signal from the original multischedule, with a larger period, will be scheduled before some new signal with a shorter period. The reader can imagine the case where the payload of the slot is 16 bits; there is only one variant with 16 signals with the one bit payload, the period equal to the hyperperiod and an empty original multischedule. In that case, the first cycle of the first slot will be filled completely, according to the first-fit policy of the scheduling algorithm, while the other cycles remain unfilled. In the second iteration, a new variant will have to be crated, which uses all the signals and, furthermore, one signal with a period equal to one communication cycle, and a payload of one bit. The algorithm will

have to allocate a new slot for the new signal during the incremental scheduling, because the first slot of the first cycle has already been filled completely. However, one slot would have been enough even in the second iteration of the incremental scheduling if the algorithm would spread the signals smartly in the first iteration, and that is why Extensibility optimization is introduced.

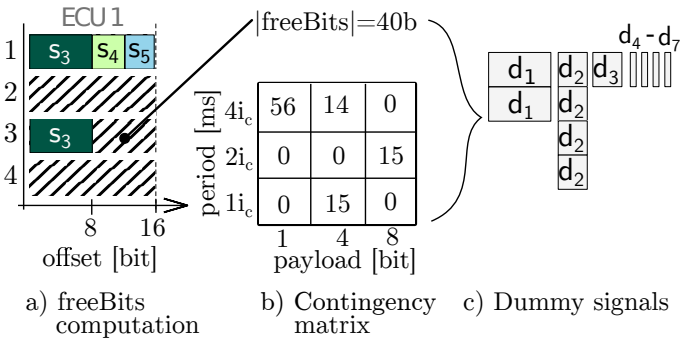

Figure 3.7: Example 3 - Process of dummy signals generation

Extensibility optimization aims to restructure the multischedule so that the multischedule remains ready for future scheduling iterations (i.e., it can accommodate as many new signals as possible) and the number of allocated slots is preserved. One way to satisfy this requirement is to perform the extensibility optimization on each slot separately. In the beginning, the algorithm computes the number of bits U in the slot (considering all cycles) that are not allocated to any signal. It means that the sum of the payloads of the signal occurrences that could be added to this slot in future is equal to U in the ideal case.

In Example 3, where only new signals s_3, s_4 and s_5 are present in the slot currently being optimized, the value U is equal to 40 bits, which is counted from the hatched part of the schedule (see Fig. 3.7.a). Moreover, the probability distribution of these parameters in the schedule is known from the p_i and c_i of the signals used in the schedule. This probability distribution is represented by a contingency table shown in Fig. 3.7.b. If the explicit parameters distribution for future signals is not given by designers, the algorithm assumes that the parameters of the future signals will follow this derived distribution.

A set of dummy signals D, whose sum of the payloads of all signal occurrences is equal to or slightly less than U and whose payloads and periods correspond to the contingency table, is generated. The generated set of dummy signals D for Example 3 is presented in Fig. 3.7.c. All signals placed in the current iteration (those that were in the SL^N list) are, consequently, removed from the multischedule of the slot and merged with those in set D to the ordered set SL^S (see Fig. 3.8.a). List SL^S is also ordered according to the procedure outlined under Sec. 3.3.4. Then the signals from SL^S are scheduled back to the slot according to the procedure given under Sec. 3.3.4.

If the resulting multischedule has only one slot, as in the one shown in Fig. 3.8.b, then the dummy signals are removed from the slot, and the extensibility optimization for the given slot is finished. Fig. 3.8.c shows the resulting structure of the slot

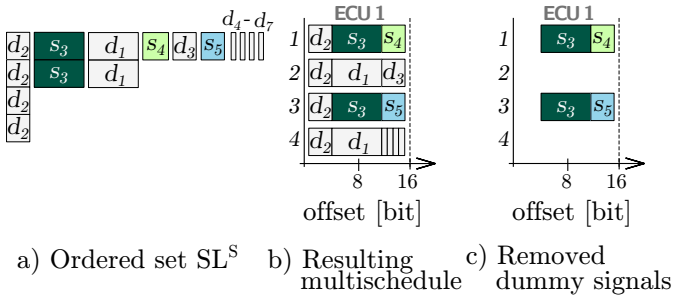

Figure 3.8: Example 3 - Slot rescheduling in the extensibility optimization

after the extensibility optimization of Example 3. Then the algorithm goes to the next slot.

Otherwise, the new U is calculated as $U_{\text{new}} = \lfloor W \cdot H - 1.05 \cdot (W \cdot H - U) \rfloor$, where W is the maximal frame payload length and H is the height of the slot (i.e., number of cycles in the hyperperiod). This formula represents the 5% decrease in the volume allocable by the new dummy signal in the slot (i.e., the total number of free bits in the slot over the entire hyperperiod). Then, the new dummy signals are generated according to U_{new}, and the scheduling is again started. This procedure is repeated until only one slot is scheduled or $U_{\text{new}} \leq 0$.

Slot scheduling

After completing Stage B for all the ECUs, the values of y_i and o_i become known for all the signals. Also, the number of slots allocated for each ECU becomes known. Then, the last step is to find the positions of the slots from the unit multischedules of the ECUs in the final multischedule. According to the problem definition, slots of no two ECUs can overlap in one variant, but those not used in the same variant (ECU 2 and ECU 3 in the original multischedule of Example 1) can overlap in the multischedule.

The slot scheduling problem is then to assign the slots, from the unit multischedules to the final multischedule, so that the number of allocated slots in the multischedule will be minimal. The problem can be formulated in terms of graph theory.

The algorithm constructs graph G_{SLOT}, where the nodes are the slots of the unit multischedules. There is an undirected edge between slots l_i and l_j, if, and only if, their transmitting ECUs $h_i = \text{ECU}(l_i)$ and $h_j = \text{ECU}(l_j)$ are both used by some variant - i.e., $EEM_{h_i,h_j} = 1$. It is to be noted that the slots from one ECU form a clique in G_{SLOT}. Moreover, the slots that are to be scheduled in one variant together form a clique. Now, graph coloring is used to solve the slot scheduling sub-problem. Each color of the resulting graph corresponds to one slot in the multischedule.

For Example 4, let it be assumed that there are five ECUs and each ECU has scheduled only one slot ($h_1 \ldots h_5$) in the unit multischedule (let the slots be labeled as $l_1 \ldots l_5$ where h_1 = UCU 1, etc.) Furthermore, assume that there are three variants. Variant I uses l_1, l_2, l_3, Variant II uses l_1, l_3, l_4, and Variant III uses l_1,

l_4, l_5. The graph G_{SLOT} for this problem is depicted in Fig. 3.9. The resulting

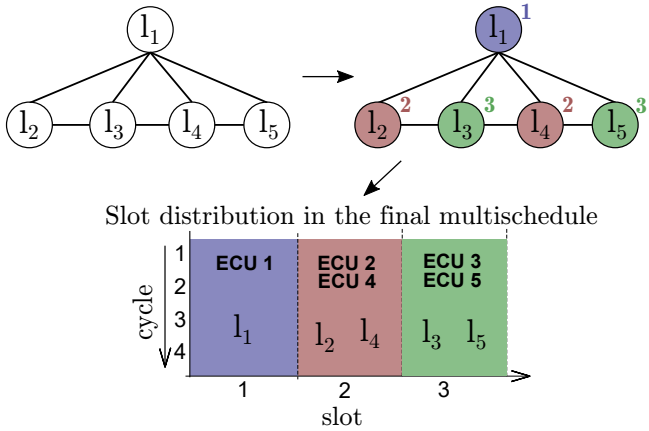

Figure 3.9: Example 4: Graph G_{SLOT} coloring

color/slotID in the final multischedule is indicated by the colored number, next to the node on the right side of the figure. In the example, the final multischedule has three slots.

The graph coloring problem is known to be NP-hard, but the heuristic algorithm can often find an optimal solution in polynomial time for slot scheduling cases. The minimum number of colors in the graph (chromatic number) must be bigger than or equal to the number of nodes in the biggest clique. It often so happens that the size of the biggest clique is equal to the chromatic number of G_{SLOT}, because in real cases cliques in graph G_{SLOT} are big. Finding the maximum graph clique is also an NP-hard problem. Fortunately, it is possible to use the number of slots in the variant, with the most slots as the lower bound, which is consequently a valid lower bound (LB) for the chromatic number too. Then, sequential heuristics is used to solve the graph coloring problem. The heuristics takes all the nodes in an arbitrary order and tries to color them, one by one, using the minimum possible number of colors. The biggest used color number becomes the upper bound (UB) for the coloring problem. If this upper bound is equal to the lower bound, then it is considered that heuristics has found an optimal solution, otherwise the bounded ILP model for the graph coloring, presented by the system of equations (3), is used. Note that the ILP model is accelerated by the knowledge (i.e., LB and UB)

obtained by the heuristics too.

$$
\begin{array}{lll}
\min & z & \\
\text{subject to} & k \cdot w_{i,k} \leq z & \forall i,k \\
& \sum_{k=1\ldots UB} w_{i,k} = L_i & \forall i \\
& w_{i,k} + w_{j,k} \leq 1 & \forall i,j,k \mid EEM_{i,j} = 1 \\
& w_{i,k} = 1 & \forall i,k \mid \tilde{w}_{i,k} = 1 \\
\text{where} & w_{i,k} \in \{0,1\}; & \forall i,k \\
& z \in [LB, UB] & (3)
\end{array}
$$

Here z represents the biggest assigned color number (i.e., slotID). Variable $w_{i,k} = 1$, if i-th ECU has the k-th slot of the final multischedule. $\tilde{w}_{i,k}$ is equal to 1, if, and only if, the i-th ECU has the k-th slot of the original multischedule and if the slot is not a conflicting one according to the *EEM*. L_i is the number of slots in the unit multischedule of the i-th ECU, and LB (UB) is the given lower bound (upper bound respectively).

If there is a conflict (as in Variant III of Example 1), the conflict is resolved by WMIS as explained in Sec. 3.3.4, with the difference that now the vertices in the conflict graph are slots rather than signals.

After slot scheduling, the full assignment $s_i \rightarrow [y_i, t_i, o_i]$ (final multischedule) becomes known for all the signals. It is also decided, whether the resulting schedule is feasible. If the number of allocated slots does not exceed the slots threshold, the schedule is considered feasible.

The proposed algorithm can be used for non-incremental scheduling too, and its results will be better than or comparable to those presented in [19].

3.3.5 Schedule feasibility

The provided algorithm tries to find the schedule with the minimum number of allocated slots. However, for given network parameters, the resulting schedule can be infeasible, because it exceeds the slots threshold. Nevertheless, if the basic network parameters are not strictly given, it could be possible to find a feasible schedule with modified configuration of the network parameters. Considering that the signal parameters are immutable, the Exploration algorithm 3.3 can modify the length of the frame or the duration of the communication cycle. The exploration algorithm enumerates all possible combinations of the network parameters and evaluates them by the above described algorithm. The results are provided to the network designer, who can choose the most suitable one. Thanks to the small computation complexity of the scheduling algorithm (see Table 3.4), the exploration can be accomplished in a reasonable time.

It is important to note here, that exploration of the network parameters is possible in the first iteration of the incremental scheduling only. Tuning would introduce a significant backward compatibility violation later on. Thus, if the resulting schedule of some later incremental iteration is not feasible, it is recommended to generate the new one by non-incremental multi-variant scheduling. On one hand, this means

Algorithm 3.3 Algorithm for the network parameters exploration

Input : Benchmark instance
Output : Number of allocated slots for different frame length and cycle duration

for *each frame payload length* $w \in \{\max c_i, \max c_i + 16, \cdots, 2048\}$ **do**
  for *each duration of the communication cycle* $m \in \{\min p_i, \cdots, \frac{\min p_i}{32}, \frac{\min p_i}{64}\}$ **do**
    Call scheduling algorithm with $W = w$, $M = m$
  end
end

the loss of backward compatibility completely, but, on the other hand, it saves both the number of allocated slots, as will be shown in Sec. 3.4.4, and the bandwidth, as the result of parameters tuning.

3.4 Experimental results

The proposed algorithm was coded in C++ and tested on a PC with an Intel®Core™2 Duo CPU (2.8 GHz) and an 8 GB RAM memory.

Eight different benchmark sets were used to evaluate the algorithm and assess the impact of the used methodologies on the resulting schedules. One of the instances used for testing was obtained from an industrial partner, and that represents a realistic case with 23 ECUs (11 ECUs are common to all variants) and more than 5000 signals. This instance was analyzed, and a probabilistic model derived to generate 30 instances, with parameters similar to those of the real case instance.

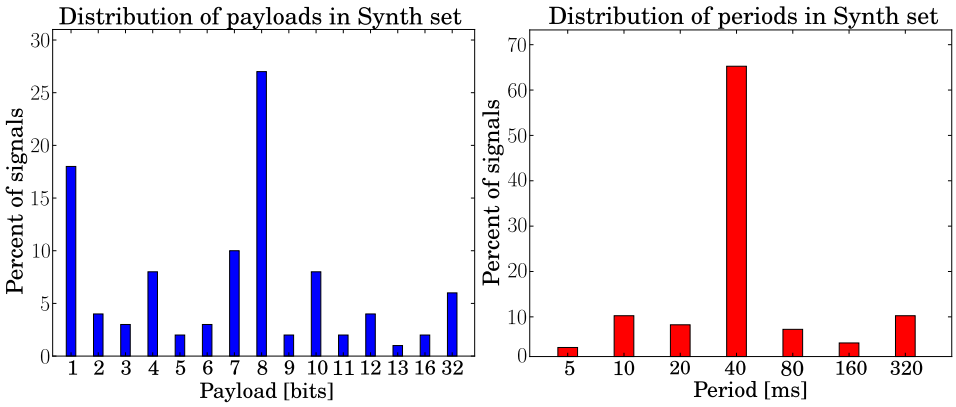

Figure 3.10: Distribution of signal parameters in Synth sets

The distributions of payloads and periods in this set are shown in Fig. 3.10. Neither release date nor deadline constraints were imposed on the signals here. These synthesized instances belong to the Synth benchmark set.

The remaining sets were based on the extended Society of Automotive Engineers

(SAE) benchmark set (originally used [26] and generated by the Netcarbench tool [10]). The signal parameters' distributions of these sets are shown in Fig. 3.11.

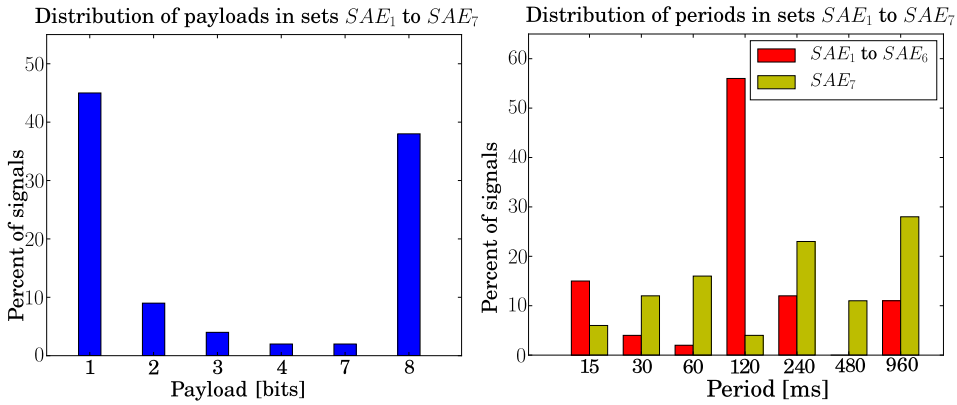

Figure 3.11: Distribution of signal parameters in SAE sets

The parameters of the instances are shown in Table 3.3, where the second column presents the number of ECUs included in the particular set. The third column presents the duration of the communication cycle, and the fourth one presents the maximum frame payload length. The fifth and sixth columns present the percentage of the signals with imposed real-time constraints. The seventh column presents the slots threshold calculated for the network configuration with the communication cycle containing only static segment and NIT with duration of 50 µs. It can be observed, from the table, that the benchmark sets SAE_1 to SAE_6

Set	ECUs [-]	L [ms]	W [bits]	Release dates [%]	Deadline [%]	Slots threshold [-]
Synth	23	5	64	0	0	176
SAE_1	3	15	32	0	0	641
SAE_2	3	15	32	25	0	641
SAE_3	3	15	32	19	19	641
SAE_4	3	15	32	40	0	641
SAE_5	6	15	64	20	0	546
SAE_6	6	15	32	20	20	641
SAE_7	23	15	32	0	0	641

Table 3.3: Parameters of individual benchmark sets

share the same periods and payloads distributions, but they differ in the real-time constraints imposed on them and the number of ECUs used in the instance. While the instances SAE_1 to SAE_4 used just three ECUs, the instances SAE_5 and SAE_6 used six ECUs. The instance SAE_7 used 23 ECUs. The portion of the signals, with imposed real-time constraints (i.e., release dates or deadlines), varies from 0 % in set SAE_1 to 40 % in set SAE_4. All the sets use FlexRay with 10 Mbit/s of bandwidth.

The *SAE* sets were designed for non-incremental, single-variant scheduling only. Therefore, the multi-variant instances for incremental scheduling iterations were generated artificially. A detailed description of the generation process can be found in Appendix 3.6. The benchmark generator and all its configuration files (one for each benchmark set) used in this study, are available in [20].

In the following subsections 3.4.1 and 3.4.2, the focus would be on non-incremental case instances, while subsections 3.4.3, 3.4.4 and 3.4.5 deal with the investigation of incremental multi-variant cases. Exploration of suitable network parameters is investigated in subsection 3.4.6 and subsection 3.4.7 concludes this section with the verification of the proposed schedules on a real FlexRay network.

3.4.1 Evaluation of various scheduling techniques for non-incremental scheduling

Different approaches to scheduling are proposed in the introductory part of this chapter. The first technique that completely prevents the problem of dissimilarities among particular vehicle variants is to create one schedule for all variants, with all the signals included. However, this technique needs the most bandwidth (i.e., it allocates the highest number of static slots). This explains why this technique is used as a reference for other related investigations. It means that the number of the allocated slots by such a common schedule (blue star) represents 100 % in Fig. 3.12.

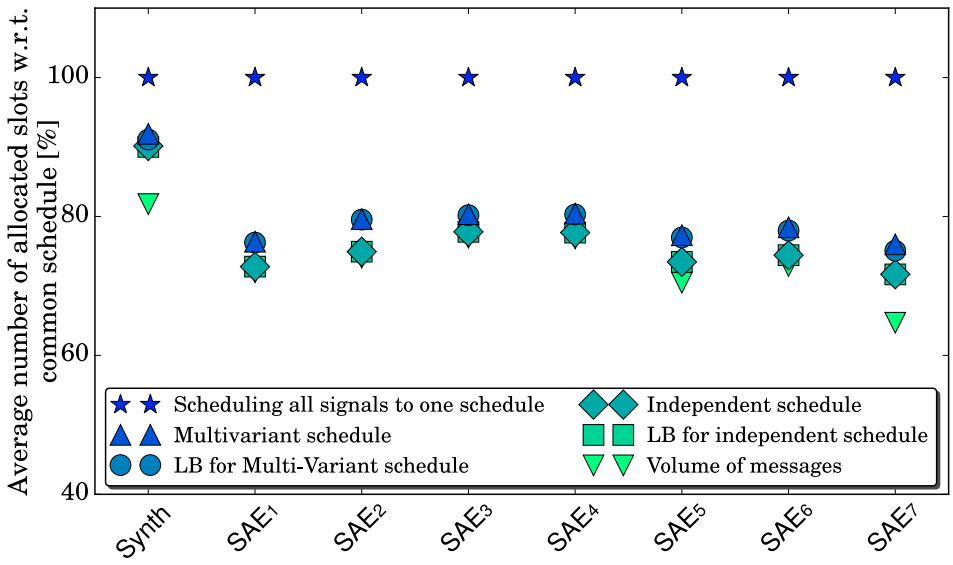

Figure 3.12: Evaluation of different scheduling techniques

Creating independent schedules (aqua colored diamond in Fig. 3.12) separately for each vehicle variant is the opposite extreme. This technique provides an ideal solution from the viewpoint of bandwidth utilization. However, this is the most unacceptable solution from the viewpoint of compatibility of variants, because one

signal is placed in different positions in different variants.

The proposed multi-variant scheduling solution, which preserves bandwidth utilization, besides sharing constraints to the maximum extent possible, is depicted as blue triangles in Fig. 3.12. The scheduling problem is NP-hard. Thus, the algorithm (Fig. 3.5) can, sometimes, miss the optimal solution as a trade-off for reduction in time complexity. That is why the lower bound values are presented in the figure. The idea of lower bound calculation is based on the lower bound algorithm for 2D bin packing. The minimal number of slots $a_{i,j}$ needed to exchange the required volume of the data through the bus is calculated independently for each ECU i and each variant j. Then, the minimal number of slots needed by ECU i in the multischedule is equal to $a_i = \max_j a_{i,j}$, because the multischedule has to contain signals from all variants. Consequently, the exact algorithm used for slot scheduling, explained under Sec. 3.3.4, is utilized to compute the feasible lower bound used here.

Moreover, the figure presents the volume of the messages (i.e., the total number of bits used by all signals over the hyperperiod, divided by the bit capacity of one slot and the number of cycles in hyperperiod) as downward pointing green triangles. Thus, the reader can evaluate the optimality gap of the heuristic algorithm.

It can be seen that the multi-variant scheduling solution needs just a little more bandwidth than independent scheduling, while preserving the sharing constraint. Compared to the common schedule, it can save about 10-30 % of the bandwidth. Moreover, the scheduling algorithm provides the solutions that are close to the lower bound (as many as 179 out of 240 solutions reached the lower bound value).

3.4.2 Evaluation of the influence of similarity on non-incremental multi-variant schedule

The parameters of instances influence the result of multi-variant scheduling. This evaluation aims to capture the sensitivity of multi-variant scheduling to the similarity of the variants. Benchmark instances, based on Synth set restricted to four variants, were created for this experiment. Three different coefficients were used to express the variants' similarity. Coefficient α represents the portion of the variant specific signals, which are included in a single variant only. Coefficient γ represents the portion of common signals, which form part common to all variants. The percentage of the shared signals (i.e., the remaining signals, which are common to two or more variants, but not to all of them) is described by coefficient $\beta = 100 - \alpha - \gamma$. The graph, showing the dependency of the number of allocated slots on these coefficients, is shown in Fig. 3.13.

If all signals are common to all variants (see left corner of Fig. 3.13), then it is enough to create one common schedule. This case naturally allocates the most slots. In the other extreme, when all signals are specific (see right corner of Fig. 3.13), the schedules can be created independently, and those schedules overlap each other. Hence, in the case of four variants, the schedules allocate almost one-quarter of bandwidth regarding the common schedule. The central corner in Fig. 3.13 represents the instances, where$\beta = 100\,\%$ ($\alpha = \gamma = 0\,\%$). Here, the number of used slots is close to one-half, as compared to those of the common schedule. The rest of the space almost represents the linear interpolation between

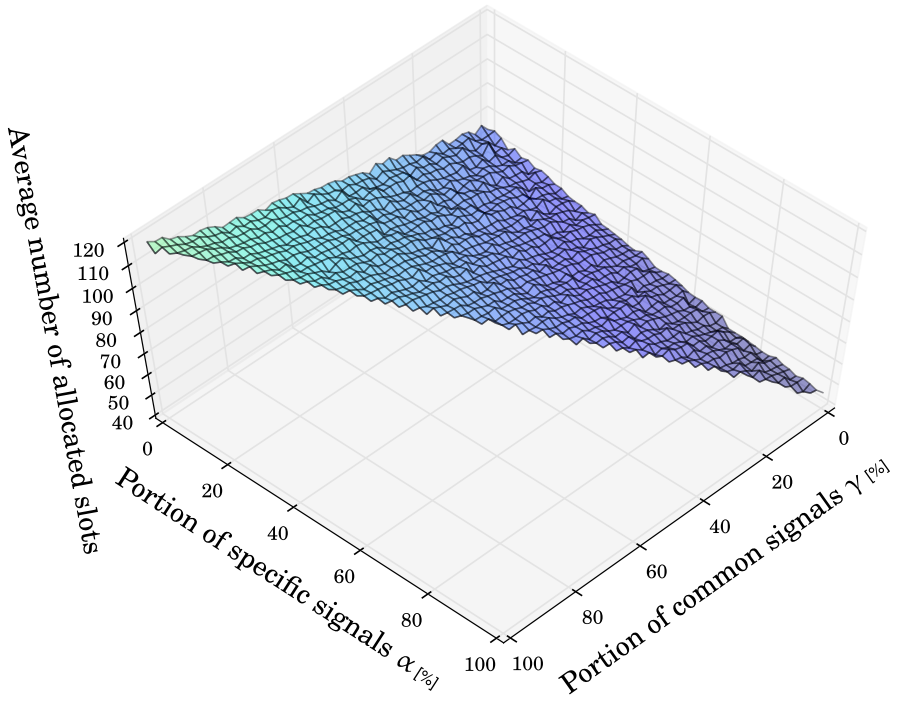

Figure 3.13: Evaluation of the non-incremental multi-variant scheduling

those three extreme cases.

3.4.3 Evaluation of the influence of similarity on the incremental multi-variant schedule

So far, all the experiments were performed for non-incremental scheduling scenarios to study the multi-variant scheduling aspect first. Therefore, the next step is to investigate how the increasing number of iterations influences the solution in incremental scheduling.

For this, let extensibility optimization be set aside for now. The multi-variant coefficients of the Synth benchmark set were simplified for this experiment so that the result can be visualized in a 3D graph. The number of common signals is equal to the number of shared signals in the benchmark set used here (mathematically expressed $\beta = \gamma = \frac{100-\alpha}{2}$) in the first iteration. For the later interations, instead of trying to preserve the multi-variant coefficients as much as possible, the new variants follow the more realistic scenario, wherein the new variant is based on the randomly choosen preceding variant. This new variant introduces new variant-specific signals and ECUs, besides introducing changes in shared signals. The common signals are preserved. This way, the results follow the real case situation.

Fig. 3.14 presents the dependence of the number of allocated slots in resulting

multischedule on the portion of common and shared signals and the iteration of the incremental scheduling. The maximum increase in the number of used slots is

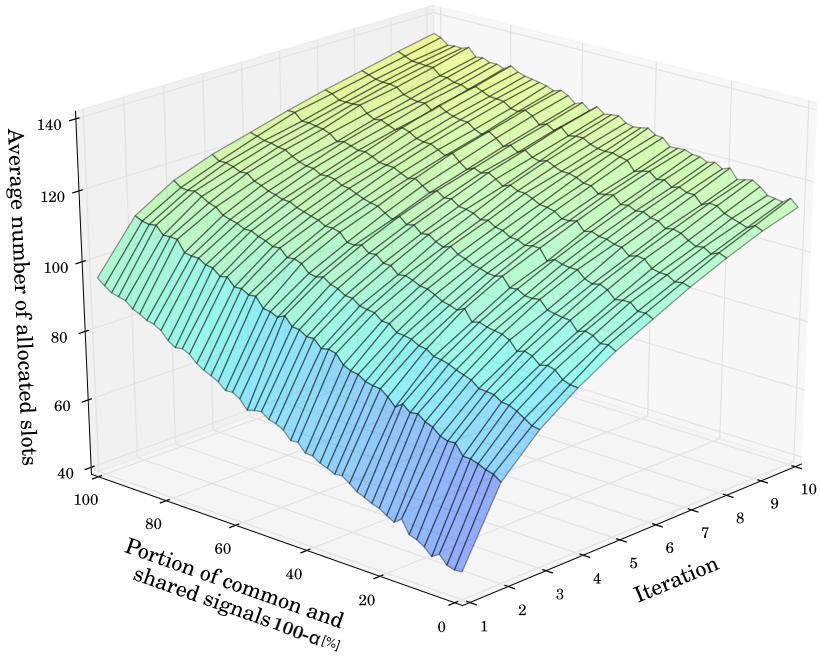

Figure 3.14: Evaluation of incremental multi-variant scheduling

between the first and the second iteration. This increase is caused by the density of the multischedule, created during the first iteration. Often, the new signal does not suit any slot allocated for a given ECU in the original schedule, and, therefore, a new slot has to be allocated for such a signal. If such a situation occurs for each ECU, the total number of slots will have to be increased by 23 (it is to be noted that the Synth benchmark set contains 23 ECUs). Those new slots introduce porosity in the multischedule, which reduces the need for allocation of new slots in subsequent iterations. One can see that the slope, which indicates the increase in the number of slots, correlates with the slope after the first iteration in Fig. 14 (it equals the line from $\alpha = 100$, $\gamma = 0$ to $\alpha = 0$, $\gamma = 50$). The slope becomes progressively gentler during subsequent iterations of incremental scheduling, because new variants cannot preserve the multi-variant coefficients.

3.4.4 Evaluation of the extensibility optimization for the incremental scheduling

The algorithm of extensibility optimization, introduced under Section 3.3.4, tries to restructure the multischedule in such a way that the probability of the algorithm needing allocation of extra slots for new signals in future is small. In the ideal case, when the extensibility optimization knows the future, the resulting incremental multischedule would be the same as the multischedule, created by non-incremental scheduling. As the algorithm does not know the future, it just tries to predict (as explained under Sec. 3.3.4). This experiment evaluates the behavior of the extensibility optimization in Fig. 3.15. In the upper part of the figure, four rows

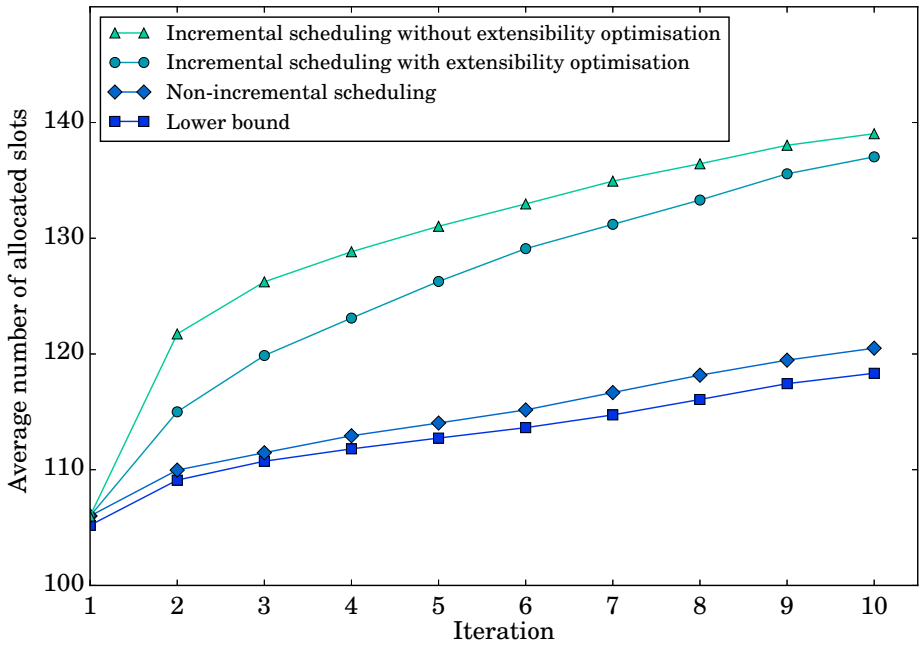

Figure 3.15: Evaluation of extensibility optimization for the incremental scheduling

are shown. Among these, the row with square marks refers to the lower bound. The row with rhomboid marks denotes the results of non-incremental scheduling for comparison of incremental versus non-incremental solution. It is to be noted that, in case of non-incremental scheduling, no backward compatibility is preserved. The row marked by solid circles and the one marked by triangles represent the results achieved for incremental scheduling algorithm, the former with extensibility optimization and the latter without it. The difference between non-incremental and incremental scheduling cases is the cost of preserving backward compatibility.

The results show that extensibility optimization is most successful during the first iteration. It follows from the scope over which the optimization can operate. While the algorithm can restructure the entire multischedule during the first iteration,

when all the signals are new, the scope becomes significantly restricted during subsequent iterations. That explains why during the last iteration, the upper two lines of the graph are close to each other. The optimization is not able to suppress the number of allocated slots to the number of allocated slots by non-incremental scheduling. This is caused primarily by two constraints: backward compatibility constraint that affects mainly late iterations, and the constraint that restricts the algorithm to keep the number of allocated slots equal to the number of those allocated in the case without optimization (recall that extensibility optimization affects the number of allocated slots in subsequent iteration and not during the current one). The second constraint is most significant in the first iteration.

It is also an important observation that in later iterations, the lines representing incremental scheduling are similar to those representing non-incremental scheduling, in terms of their slope. Thus, the scheduling can utilize the porosity in the schedules efficiently.

3.4.5 Evaluation of Incremental Multi-variant scheduling algorithm

This section focusses on a comprehensive evaluation of the performance of the proposed algorithm, in contrast to previous evaluations, which focused only on the behavior of incremental and multi-variant scheduling, and aims to present the results in a precise form.

The evaluation-sets follow the parameters' distribution, as described under the introduction of Sec. 3.4. The instances contain more than 5000 signals in the first scheduling iteration and more than 6000 in the last one.

	Iteration									
Set	1	2	3	4	5	6	7	8	9	10
Synth	105.7	114.3	118.9	121.9	124.5	127.9	130.3	132.2	134.7	136.5
SAE_1	122.8	139.6	143.9	147.2	149.7	153.4	156.5	160.3	163.9	167.3
SAE_2	131.5	143.7	147.8	151.1	154.7	158.2	161.3	164.0	167.2	170.2
SAE_3	131.6	144.9	149.4	152.6	156.6	160.3	162.8	165.2	168.4	170.9
SAE_4	132.0	142.9	146.9	150.3	153.0	156.0	158.7	161.5	164.0	167.1
SAE_5	64.8	75.2	78.2	80.6	82.9	84.9	86.6	88.4	90.1	91.7
SAE_6	127.1	145.9	151.4	155.0	158.6	161.8	165.1	168.3	171.5	174.6
SAE_7	99.3	120.6	127.3	133.0	136.9	140.4	143.5	147.4	150.6	153.6
Ex. time [ms]	314.0	20.0	16.6	17.3	16.2	18.7	19.4	17.4	19.0	18.6

Table 3.4: Number of slots and execution time of incremental multi-variant scheduling algorithm on different sets

The results of the algorithm are presented in Table 3.4. In this table, the row of the cell determines the set, and the column the iteration of incremental scheduling. Each cell presents the number of allocated slots in the multischedule. The value is averaged over all the instances in the set. The last row shows the execution time of the algorithm, for the given iteration, averaged over all the benchmark instances. The first iteration has been the slowest one, because it has to place the biggest number of signals; besides, the conflict graph also is mostly much larger. Even

though, the execution time in hundreds of milliseconds for industrial sized instances is incomparable with a development cycle of any vehicle variant.

3.4.6 Exploration of the network parameters

In order to evaluate the influence of the network parameters, an extra benchmark instance, for which the number of allocated slots in the resulting schedule reaches the slots threshold, was created. The duration of the communication cycle is 8 ms in the instance. Similarly, the minimum period $\min p_i$ is also 8 ms and, hence, the evaluated durations of the communication cycle are 8, 4, 2 and so on down to $\frac{1}{8}$ ms. The instance contains more than 25000 signals. The signals follow the signal parameter distribution as signals in the Synth benchmark set, but the signal periods were changed from a 5 ms scale to an 8 ms scale. The network parameters were enumerated and evaluated by Algorithm 3.3 and the results are presented in Fig. 3.16. In the figure, each point represents one combination of the network parameters together with the resulting schedule. The figure shows three data

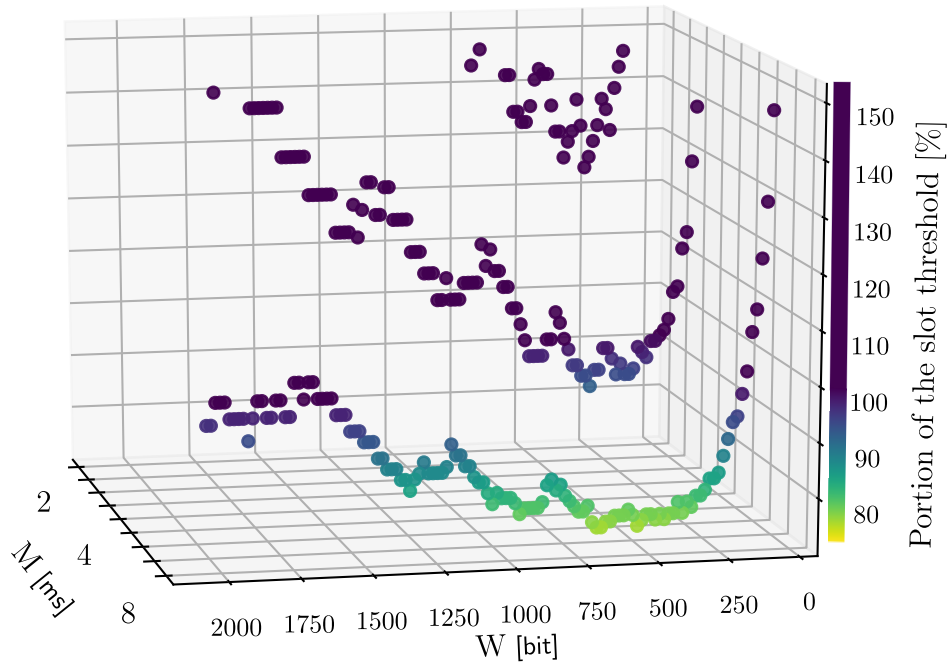

Figure 3.16: Influence of the network parameters on the efficiency of the resulting schedule

rows, where each data row represents one configuration of the duration of the communication cycle. Only three different values for communication cycle duration are shown in the figure to simplify the readability. Note that the modification of the network parameters does not only influence the number of slots in the resulting schedule, but it also influences the slots threshold. Thus, Fig. 3.16 presents the

number of allocated slots as a percentage of the slots threshold rather than as a number directly. The percentage is also represented by the color of each point, where all the points with dark blue color are over 100 %. It loosely corresponds to the portion of the communication cycle used by the static segment, taking into account that NIT is minimal and the Dynamic segment and Symbol window are not used.

The length of the frame is bounded from the bottom by the payload of the longest signal (which is 32 bits in our case). Similar limitation holds for the duration of the communication cycle which is bounded from the top by the signal with the smallest period.

It can be observed from Fig. 3.16 that the decrease in the duration of the communication cycle causes the increase in the allocated portion of the communication cycle. The signals with the longest period (e.g., 512 ms in our case) must be transmitted with the shorter period (e.g., 256 ms, if the duration of the communication cycle was decreased from 8 ms to 4 ms), which causes the signal retransmissions and, consequently, the increase in the allocated portion of the communication cycle. Thus, this modification often does not solve the problem with the infeasibility of the resulting schedule.

On the other hand, the modification of the length of the frame can significantly decrease the allocated portion of the communication cycle. If the length of the frame is prolonged, then the bandwidth of the bus is used more efficiently because fewer macroticks are consumed by, for example, the inter-ECU synchronization mechanisms (action points), etc. The prolongation is efficient as long as the number of allocated slots is strictly greater than the number of ECUs. If the resulting number of allocated slots is small, the overhead of the non-filled slots overwhelms the gained efficiency.

3.4.7 Verification of the resulting schedules on hardware

The last step in evaluation is to verify the feasibility of the resulting schedules, for which, two methods were used. The first method utilizes the feasibility validator, which goes through all the hard constraints, derived from the communication protocol (in the present case FlexRay protocol), multi-variant and incremental scheduling, and then checks the validity of their results. The advantages of validator are its versatility and efficiency, because they can handle a huge number of instances in a matter of seconds. This algorithm was used to check all the schedules used for the present evaluations. However, the validator cannot check all the hardware-related constraints and parameters, because its point of view is at too high a level (it just checks the correctness of schedules with respect to the mathematical model of the bus, and not the real bus). While deploying the schedule to the real network, the low-level parameters (such as duration of macrotick- and microtick-relating to the bandwidth used, the number of macroticks in communication cycle, duration of static slot in the number of macroticks, duration of static segment, static slot, symbol window, and network idle time, besides more than 80 other parameters - for more details the reader can refer FlexRay specification [29]) will have to be properly set to obtain the functional solution. For this reason, the second method - verification of the resulting schedules on hardware - was also included.

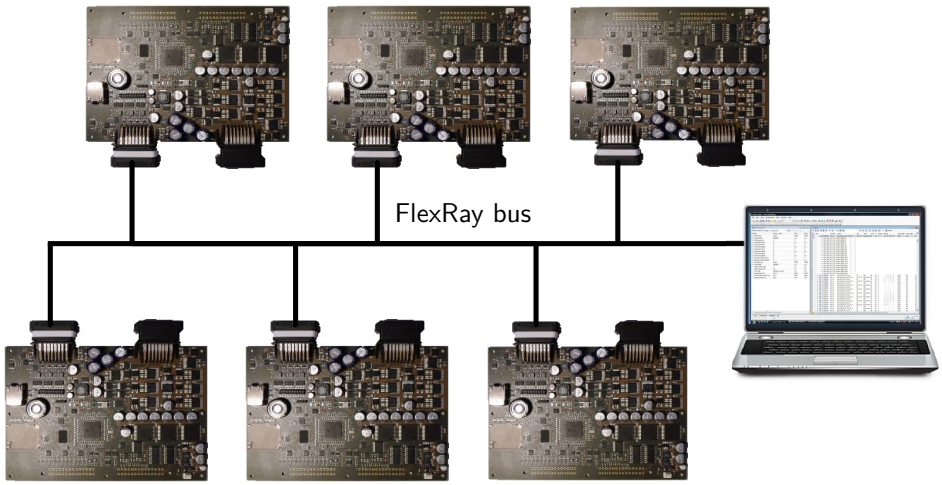

Figure 3.17: The block diagram with wiring of the evaluation system

For the testing purpose, the authors used a system of six ECUs, represented by Rapid Prototyping Platform boards, constructed in their labs [27], which were interconnected with FlexRay bus (see Fig. 3.17). The bus was connected to a notebook with FlexRay analyzer and Vector CANoe software [65] for capturing and examining the communication. Figure 3.18 presents a photograph of the system used.

The scheduling algorithm provides the resulting schedules and all the network configuration parameters in FIBEX database format [3]. CANoe can read the database and accordingly parse the captured communication. Such a link between scheduler and analyzer facilitates easy verification of the communications happening on the bus. Moreover, CANoe also provides the counters for erroneous frames (e.g., frames, which do not follow the schedule as presented in FIBEX database).

A new set of instances was used for verification, because the number of ECUs was only six. The signal set was generated in such a way that the communication in the static segment almost covers the full bandwidth. The payload data was set to '1' for each signal. It allows distinguishing the signals in a plain stream from analyzer (the value '1' in the stream serves as the delimiter of messages), even when the FIBEX database is not used. The firmware generator, which takes the schedule for a given iteration of given variant and generates the code in C for each board involved, was implemented. The code was compiled and uploaded to the corresponding board.

The second method is more rigorous, but is much more time-demanding. As it is necessary to analyze the schedule for each variant and each iteration, independently, this method takes hours to go through the process for just one instance. Moreover, this method cannot check if the constraints, relating to multi-variant and iterative scheduling (e.g., sharing constraint and backward compatibility constraint) are satisfied. Therefore, both methods were used to demonstrate the correctness of the

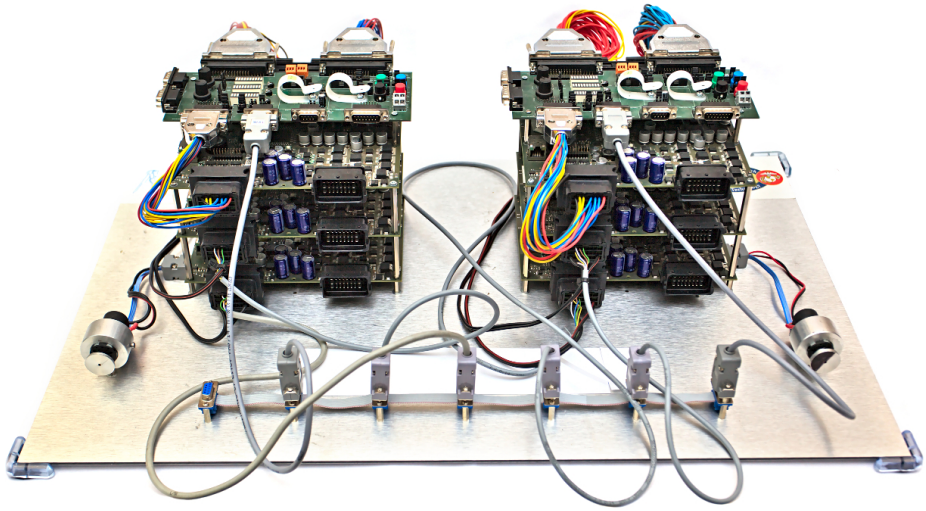

Figure 3.18: FlexRay system with six Rapid Prototyping Platform boards

proposed algorithm.

3.5 Conclusion

This chapter tackles the problem of scheduling of the time-triggered internal vehicle communication for multiple vehicle variants. It presents the solution where the shared constraint among variants is preserved, while optimizing the utilization of the bandwidth. Moreover, the proposed solution takes the incremental iterations of variant development into account and minimizes the number of backward compatibility violations. It also uses an extensibility optimization heuristic, which tries to predict future signals of the following design iteration and enhances the schedule, such that it allocates less bandwidth subsequently. The results of the algorithm were verified on the FlexRay bus system to prove the validity of the concept. However, the described methodology is not restricted to FlexRay.

The experimental results are discussed, focusing on the analysis of dependence of the bandwidth demands on the multi-variant and incremental scheduling paradigm. Besides, the bottlenecks and limitations are also pointed out. The linear relation between the similarity parameters of variants and the resulting number of allocated slots in the schedule shows the advantages of the multi-variant approach. The relation between bandwidth occupancy and the iteration of incremental scheduling appears to be more complicated, which, among others, is the consequence of the impossibility of correct prediction. The algorithm was evaluated on SAE group with real-case inspired instances, demonstrating that its performance complexity is negligible. The used instances are accessible in [20].

3.6 Appendix - Benchmark instance generation procedure

In this section, the process of the multi-variant benchmark instance generation is described. The process is depicted from a high-level in Algorithm 3.4. At the

Algorithm 3.4 Multi-variant benchmark instance generation process

Input : Instance parameters
Output : Multivariant benchmark instance

Read the instance parameters
for *each signal* s_i *in* S **do**
    Generate the signal period
    Generate the signal payload
    Generate the signal deadline
    Generate the signal release date
end
Assign the transmitting ECU to commom signals
Assign the transmitting ECU to specific signals
Assign the transmitting ECU to other signals
Generate variant matrix $V_{i,j}$
Repair instance $V_{i,j}$

beginning, the required instance parameters are read. These parameters consist of distributions presented in Fig. 3.10 or Fig. 3.11, parameters from Table 3.3, the number of signals and the number of variants to generate, the multi-variant coefficients α and β for signals and similar coefficients for ECUs. Note that ECUs can be common to all variants (so-called common ECUs) or specific to just one variant (so-called specific ECUs) in the same way as signals can be.

Subsequently, the basic signal parameters are generated for each signal. The periods and payloads follow the requested distributions. Deadlines and release dates are generated only for the requested portion of the signals determined by the instance parameters. The deadline is set to the end of a randomly chosen communication cycle from the last third of the signal period. The release date is set to the beginning of the communication cycle also randomly chosen from the first six communication cycles.

After generation of basic signal parameters, the transmitting ECUs are assigned to the signals. Firstly, the ECUs are assigned to the common signals. The common signals can be transmitted from the common ECUs only. The common ECUs similar to the common signals have to be included in all variants. Secondly, the ECUs are assigned to the specific signals. Inversely to the case with common signals, specific ECUs are allowed to transmit specific signals only. Otherwise, the specific ECUs would be forced to appear in more than one variant. In the end, the ECUs are assigned to the rest of the signals that are shared, and it is assured that each ECU transmits at least one signal.

The generation of variant matrix $V_{i,j}$ is divided into two steps. The first step is deciding which ECUs are used in which variant. The common ECUs are used in all

variants, and the specific ECUs are used only in one randomly chosen variant. The rest of the ECUs are distributed to the random subset of variants. In the second step, the signals are assigned to the variants. All the common signals are assigned to all the variants. The specific signals that are transmitted by the specific ECUs are assigned to the same variant as the specific ECUs. The rest of the specific signals are assigned to randomly chosen variants to which the transmitting ECU is assigned. For the case of shared signals, random probability from 30 to 70 % is chosen for each variant. This probability determines whether it is rather luxurious or economy variant. With this probability, the shared signals are assigned to the particular variant if the transmitting ECU is used in the variant.

According to this strategy of assigning signals to variants, situations can occur when some signal is not assigned to any variant. Thus, it is necessary to repair such issues in matrix $V_{i,j}$. Each signal in $V_{i,j}$ is checked whether the signal is assigned to some variants. If it is not, the signal is assigned to a random subset of variants assigned to its transmitting ECU. Finally, the admissible multi-variant instance is generated that satisfies all the requested instance parameters.

However, the described process does not take into account any predecessor benchmark instance and, thus, it is useful only for the generation of the benchmark instance for the first iteration of incremental scheduling. The generation of subsequent iterations follows a process depicted in Algorithm 3.5. The generation starts

Algorithm 3.5 Generation of the consequent interations of benchmark instances for the incremental scheduling

Input : Instance parameters
Instance for the previous incremental iteration
Output : Incremental multivariant benchmark instance

Read the instance parameters
Read the instance for the previous incremental iteration
for *each new signal* s_i *in* $S \setminus \tilde{S}$ **do**
  Generate the signal period
  Generate the signal payload
  Generate the signal deadline
  Generate the signal release date
end
Assign the transmitting ECU to the new signals
Add the new variant to variant matrix $V_{i,j}$

with the reading of the requested instance parameters. Then, the instance of the previous incremental iteration is read. The new instance is going to be based on this so-called original instance. All the signals and ECUs from the original instance will be present in the new instance with the unchanged basic parameters.

Then the basic parameters are generated for each new signal. The generation process is the same as in case of non-incremental instance generation. However, in this case, it cannot be assured that the new instance will follow the requested instance parameters because the parameters distribution in the original instance can vary significantly from the requested instance parameters.

In the next step, the new signals are assigned to its transmitting ECUs. If

there is no new ECU, then the signals are uniformly distributed among all ECUs. However, if there are some new ECUs, 70 % of the new signals are distributed among these new ECUs, and the rest is uniformly distributed among all the ECUs. Moreover, it is assured that all new ECUs are used in the new instance.

Finally, a new variant is added to the variant matrix $V_{i,j}$. No variant used in the original instance is changed. The new variant is based on a randomly chosen variant (so-called original variant) from the original instance, and all new signals and new ECUs are assigned to it. Because it is not often the case, in practice, that the new variant just adds new signals and ECUs to the original one, part of the variant matrix $V_{i,j}$ copied from the original variant is mutated. The signals from the original instance are processed one by one. Each signal has a 70 % chance that it will not be passed to the mutation stage at all. Once the signal reaches the mutation stage, the following mutation rules are employed:

- If the signal appears in the original variant, it has a 35 % chance that it will not appear in the new variant.
- If the signal does not appear in the original variant and its transmitting ECU appears in original variant, it has 65 % chance that it will appear in the new variant.
- If the signal and its transmitting ECU does not appear in the original variant, it has $\frac{1}{3}$ % chance that it will appear in the new variant. In this case, the transmitting ECU is added to the original variant also.

After all these steps, the new incremental multi-variant benchmark instance is ready.

Chapter

Incremental scheduling of the Time-Triggered traffic on TTEthernet network

4

4.1 Abstract

Complex systems are often developed incrementally when subsequent models must be backward compatible with the original ones. This requirement is relevant not only to particular components of the system, but also to the technology that interconnects them. The need to exchange high-volume data, for example, multimedia streams for infotainment in the avionic systems, together with safety-critical data, puts demands on both the high bandwidth and the deterministic behavior of the communication. TTEthernet is a protocol that has been developed to face these requirements while providing the generous bandwidth of Ethernet with up to 1 Gbit/s and enhancing its determinism by enabling the transmission of the time-triggered messages. However, the efficiency of the time-triggered communication depends on the schedule it follows. Thus, synthesizing a good schedule that meets all the real-time requirements and preserves the backward compatibility with the schedules of preceding models is essential for the performance of the whole system.

In this paper, we study the problem of designing periodic communication schedules for time-triggered traffic. The aim is to maximize the uninterrupted gap for the remaining non-real-time traffic. The provided scheduling algorithm, based on MILP and CP formulation, can obtain good schedules in a reasonable time while preserving the backward compatibility. The experimental results show that the time demands of the algorithm grows exponentially with the number of messages to be transmitted, but, even for industrial-sized instances with more than 2000 messages, the algorithm is able to return the close optimal schedules in the order of hundreds of seconds.

4.2 Introduction

The development process in many industrial fields, e.g., automotive or avionics, is based on incremental steps where new models are an evolution of the previous ones. This incremental process enables the cost-efficient development for companies and reduces the test effort. Moreover, the customer is guaranteed that the new model is an upgraded version of the model that he or she is comfortable with. These benefits are a side effect of the backward compatibility that should be assured among incremental development steps. Backward compatibility affects external systems, e.g., human-machine interface, as well as internal ones such as the communication subsystem. The backward compatibility in communication subsystems significantly reduces the costs spent on debugging, testing, and maintenance as newly developed

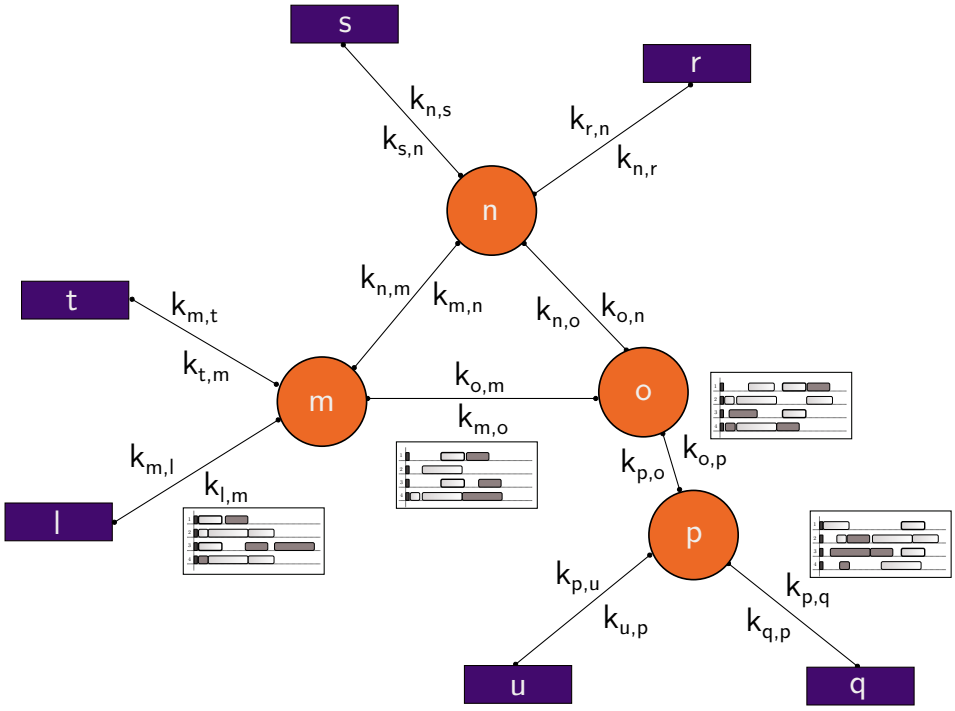

Figure 4.1: An example of the TTEthernet network topology with the routing and scheduling of message m_1 from node l to node q

Electronic Control Units (ECUs) and diagnostic tools can follow an agreement on sharing of communication resources achieved in previous development steps.

The incremental development process is already ingrained in the industrial practice. However, there is an ongoing effort to develop and produce new models even more cost-efficiently. One possibility on how to reduce production costs in complex interconnected systems is to combine safety-related communication together with non-critical communication into one common medium [38]. Safety-related communication requires determinism, while non-critical communication demands a huge bandwidth without hard timing constraints. In the past, these two communication flows were conducted separately, as there were no communication protocols that could handle the requirements of both. However, modern protocols, like TTEthernet, were developed to bear such a difficult task.

In TTEthernet, safety-related communication is exchanged based on a periodic time-triggered communication schedule, while non-critical communication fills the empty gaps in the schedule. Such a communication schedule has to be designed in a way that all the real-time requirements are met to enable the reliable and deterministic operation of the application. The creation of the schedule involves additional complexity compared to the bus or passive star topologies of networks

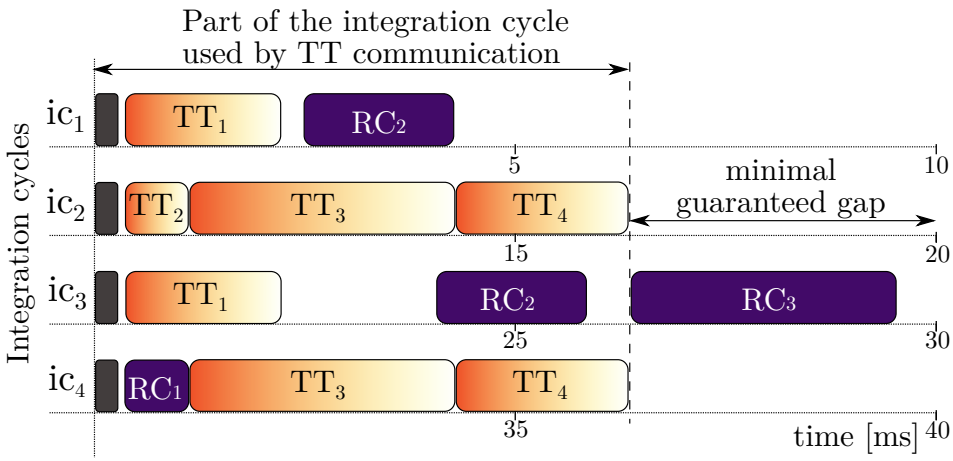

Figure 4.2: The example of the communication on a link in one cluster cycle

like FlexRay [29] or CAN because TTEthernet supports complex switched topologies [64]. These aspects, together with the real case problem proposed by our avionics industry partner, have motivated us to face the problem of scheduling time-triggered communication on the TTEthernet network while keeping the incremental development process in mind.

The paper presents the algorithm for creating schedules for time-triggered traffic on the TTEthernet network while maximizing the minimal guaranteed continuous gap for the traffic with lower criticality. The study aims to develop the periodic scheduling algorithm, which preserves the backward compatibility with the original schedule. Finally, the influence of the backward compatibility on the communication schedule is analyzed, and the scalability of the proposed algorithm is presented.

4.2.1 TTEthernet Overview

TTEthernet (TT stands for Time-Triggered) is an extension of Ethernet for deterministic communication developed as a joint project among the Vienna University of Technology [34], TTTech, and Honeywell, and standardized as SAE AS 6802 [47] in 2011. It operates at Level 2 of the ISO/OSI model, above the physical layer of Ethernet. It requires a switched network with full-duplex physical links, such as Automotive Ethernet standard 1000BASE-T1. An example of the TTEthernet topology is depicted in Fig. 4.1.

The global time in the system is assured by the clock synchronization protocol, where the clocks of all the interconnected ECUs are being synchronized periodically. Every synchronization period is called an *integration cycle*.

The traffic with various time-criticality is integrated into one physical network. There are three traffic classes in TTEthernet. These classes, ordered by decreasing priority, are Time-Triggered (TT), Rate-Constrained (RC) and Best-Effort (BE) traffic.

The TT traffic class has the highest priority. A jitter shorter than µs can be

achieved on a physical layer (the physical layer jitter also depends on the connected network devices). The TT messages are periodic. We assume that they are strictly periodic (i.e., no jitter in application level is allowed) in agreement with [62]. The least common multiple of their period is called the *cluster cycle*.

For traffic with less strict timing requirements, the RC traffic class can be used. This traffic class conforms to the ARINC 664p7 specification [2] (also called AFDX). The RC traffic represents event-triggered communication, which does not follow any schedule known in advance.

A simple example of the TT traffic, together with the RC traffic on one direction of a physical link, is presented in Fig. 4.2. In the figure, the particular integration cycles are situated in rows, and the horizontal axis represents the time instants in the particular integration cycle. The length of the cluster cycle (40 ms) is equal to four times the length of the integration cycle (10 ms) here. The figure shows that messages TT_1, TT_3 and TT_4 have the same period (twice the duration of the integration cycle - i.e., 20 ms) and TT_2 has a period equal to four times the integration cycle length. The dark message at the beginning of each integration cycle is the synchronization message.

Standard Ethernet traffic can be transmitted through the network too. Such traffic is called the Best-Effort (BE) traffic and has the lowest priority.

When the TT traffic is used together with other traffic classes, a TT message could be delayed by another RC or BE message. The delay happens when a TT message arrives while an RC or BE message is in transmission. The Timely block integration policy, which causes no extra delay of the TT traffic, is used in this paper. In this case, an RC or BE message can only be transmitted if there is enough time for the transmission of the entire message before the next TT message is scheduled. If there is insufficient time, the transmission of the RC or BE message is postponed until after the TT message is transmitted. It additionally means that the TT traffic follows the schedule without any delays.

4.2.2 Related works

The area of time-triggered communication scheduling on Ethernet-based networks has already been examined in many publications. Steiner [57] was among the first to study the problem. They described the basic constraints for scheduling the communication in the TTEthernet network and provided the Satisfiability Modulo Theories (SMT) formulation that was able to find a feasible schedule for small instances with up to 100 messages. The concept of schedule porosity was introduced in [58]. The porosity (the allocated blank slots for RC messages spread over the integration cycle) is introduced to the schedule to decrease the delay posed on RC traffic by T traffic. Thus, the porosity allows improving the communication delays while the complexity of the direct delay optimization [52] is tackled. To evaluate the impact of the porosity on the RC traffic, Steiner et al. provided a pessimistic worst-case delay calculation, which was consequently tightened by a new method published by Tamas-Selicean et al. in [62]. A more detailed study of the impact of the time-triggered schedule on the RC communication has been presented in [9]. In [61], Tamas-Selicean et al. employed the TabuSearch algorithm to overcome the scalability problem of previous SMT formulations. As noted by [59],

porosity scheduling has a disadvantage that gaps introduced at the beginning of the scheduling process do not consider the profile of the RC traffic. The concept of porosity is also weak in the case of scheduling TT messages with short periods. Wang et al. [68] used back-to-back schedule optimization, which aims to minimize the standard deviation of the messages offset in the integration cycle (hence, create as compact schedule as possible), to overcome the weakness of the porosity approach. The concept of minimization of the TT communication block length, called makespan minimization, was presented by Dvorak et al. in [21]. The paper formulated the scheduling problem as an RCPSP model to solve the problem efficiently. However, their method did not allow one to preserve the backward compatibility, and the quality of the resulting schedule was limited by the use of naive shortest-path-tree routing algorithm. Pozo et al. in [45] used a divide-and-conquer method to overcome the scheduling scalability limitations in large-scale hybrid networks considered to be used in, for example, smart cities in the future.

Based on the given TTEthernet communication schedule, Craciunas et al. scheduled the tasks on the communication endpoints in [17]. Furthermore, they presented a holistic scheduling algorithm that makes network-level schedules together with task-level schedules in [14]. Zhao et al. studied the problem of holistic security-aware scheduling in [72]. They used a modified TESLA authentication mechanism to protect the authenticity of the messages and provided MILP-formulation based scheduling algorithm.

The closely related problem to TTEthernet scheduling is the scheduling of TT communication for IEEE 802.1Qbv, which is the standard of the IEEE Time-Sensitive Networking group. Craciunas et al. derived the scheduling constraints for the TT communication on IEEE 802.1Qbv in [16] and provided an SMT model that aims to minimize the number of queues needed to schedule a given set of messages. Consequently, Zhao, together with Pop and Craciunas, provided the calculus for the Worst-case delays in [71]. Rottenstreich et al. [46] are using a greedy algorithm to find the shortest schedule for strictly periodic data streams and show that the greedy algorithm is able to find the optimal solution in special cases that often occur in practice.

All the published papers aim to create schedules from scratch, and none of them considers backward compatibility with the preceding systems, which limits the use of the proposed method in industries with an incremental development process.

4.2.3 Contribution and paper outline

The main contributions of this paper are:

1. The formal description of the incremental TTEthernet scheduling problem with real-time constraints.

2. The three-stage heuristic algorithm, which includes

   - the routing algorithm that balances the communication load among the links
   - the message-to-integration cycle assignment algorithm that balances the communication load among the integration cycles

- the message scheduling method based on the constraint programming model of the problem

3. An examination and discussion of the impact of the incremental aspect on TTEthernet scheduling.

4. An evaluation of the proposed algorithm from quality and performance point of view.

The paper is organized as follows: Section 4.3 describes the studied problem of the incremental TT message scheduling in the TTEthernet network comprehensively. In Section 4.4, the proposed method of the schedule creation is described consisting of a message routing method, a load-balancing heuristic, and a CP based formulation of the scheduling problem. The method and the impact of the backward compatibility on the scheduling are evaluated and discussed in Section 4.5. Section 4.6 concludes the paper.

4.3 Problem statement

This paper aims to design a method for finding feasible strictly periodic schedules for time-triggered communication on the TTEthernet network so that the maximal part of the remaining bandwidth can be preserved for the RC and BE messages, the timing constraints are satisfied, and the backward compatibility with the original schedule is preserved. All aspects of the tackled problem are described in this section.

4.3.1 Messages

Each message m_i from a set of the TT messages M that is to be scheduled has the following parameters:

- p_i - period
- c_i - message length in the number of bits consisting of a payload, headers and interframe gap
- d_i - deadline
- r_i - release date
- q_i - identifier of the transmitting node
- Q_i - set of the receiving nodes identifiers (the set contains only one receiving node in case of a unicast message)

The message period p_i is assumed to be an integer multiple of the length of the integration cycle *ic*. The length of the resulting schedule is determined by the length of the cluster cycle *cc*. The cluster cycle consists of set of the integration cycles I. The transmission time of message m_i has to be smaller than or equal to the duration of the integration cycle (it would not be possible to send a synchronization

message otherwise), and its length c_i does not exceed the maximal Ethernet frame length of 1530 bytes. Deadline d_i and release date r_i are assumed to have the value in the range $0 \leq r_i \leq d_i \leq p_i$.

4.3.2 Network topology

The TTEthernet topology consists of nodes and links which interconnect them. The nodes $e_i \in E$ are divided into two classes: *redistribution nodes* E^R and *communication endpoints* E^C. The communication endpoints are nodes that generate or process the data (e.g., sensors, actuators, control units, and other ECUs). Thus, only the identifier of a communication endpoint can be assigned to message m_i as transmitter q_i or one of the receivers from set Q_i. The redistribution nodes, on the other side, are switches without any of their own data to transmit and serve as intermediary nodes for the communication. In Fig. 4.1, the communication endpoints are titled by "ECU", and the redistribution nodes have arrows drawn on the top side. The front side of each node is labeled by its name.

Each hop in the network introduces a technical delay caused by queuing in the ingress and egress port. Such a delay in a switch is represented by parameter τ for the TT messages. The value of τ can be in the range from 1 µs to 2.4 µs according to the network configuration [56].

Each link $k_{i,j}$ from a set of links K connects two nodes e_i and e_j. This connection covers just one direction of the full-duplex communication. Therefore, two links $k_{i,j}$ and $k_{j,i}$ model one full-duplex physical link between nodes e_i and e_j. These two links are two independent resources from the scheduling point of view. The instance of message m_i in link $k_{l,m}$ is called a *message instance* $m_i^{l,m}$. The set of all the message instances is denoted by MI. All the transmissions of some message m_i in one particular link represent the same message instance. The *message occurrence*, on the other hand, represents all the transmissions of some message m_i in one particular integration cycle. The difference between the message instance and the message occurrence is graphically explained in Fig. 4.3. The figure shows the detailed view on the sub-segment of the network topology from Fig. 4.1 with node e_l, e_m and e_o only. Both links of any physical link are labeled here already.

4.3.3 Message routing

A sequence of the links $R_i^q = (k_{l,m}, k_{m,o}, ..., k_{p,q})$ represents the routing path of message m_i from transmitter $q_i = e_l$ to receiver $e_q \in Q_i$ through the redistribution nodes $e_m, ..., e_p$. The union of all the routing paths $\cup R_i^q \mid \forall q \in Q_i$ for given message m_i determines the routing tree R_i. For example, the transmission of message m_1 through routing path S_1^q is presented in Fig. 4.1. Only one direction of each physical link is labeled in the figure for the sake of simplicity. The routing paths R_i^q are not known in advance. Therefore, finding the appropriate routing trees is part of the optimization process.

4.3.4 Original schedule

Additionally, the original schedule is given for the incremental scheduling. The original schedule defines the start time of transmissions for the message instances

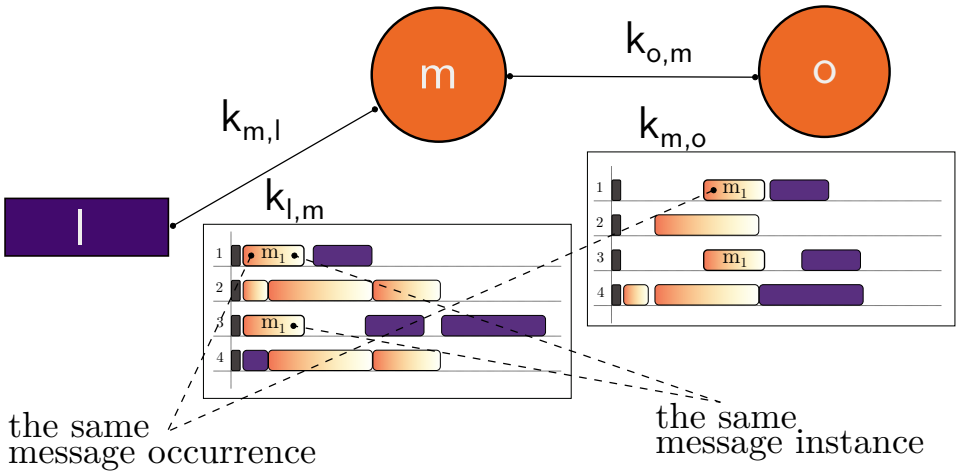

Figure 4.3: Visualization of the difference between message occurrence and message instance

$\widetilde{m}_i^{k,l}$ from the subset of the message instances $\widetilde{MI} \subset MI$ which are already present in the original schedule. The original schedule can be determined for a subset of messages as well as for a subset of its message instances. This representation of the original schedule even allows the extension of the topology as well as the extension of the set of receivers for the already present messages. More precisely, all the modifications of the topology and message set that do not force the backward compatibility to be broken are allowed in the incremental development process.

4.3.5 Schedule

The schedule is so-called strictly periodic, which means that the next message occurrence of message m_i in a particular link appears in the schedule exactly p_i time units after the current one. Therefore, the positions of all the message occurrences of message m_i in the strictly periodic schedule can be deduced from the position of the first message occurrence and its periodicity.

A feasible schedule has to fulfill the following hard constraints:

Completeness constraint: Each message $m_i \in M$ has to be scheduled.

Contention-free constraint: Any link is capable of transferring at most one message at a time.

Timing constraint: Each message has to be transmitted after its release date and received by all the receivers before its deadline.

Transmission compactness constraint: The message transition from transmitting node q_i to all the receivers from Q_i has to be accomplished in one integration cycle.

Backward compatibility constraint: The start of transmission time must be preserved for all the message instances participating in the original schedule.

Precedence constraint: Message m_i has to be scheduled in link $k_{m,o}$ at least τ

time units after it is scheduled in $k_{l,m}$ if $k_{l,m}$ precedes $k_{m,o}$ in R_i^q.

4.3.6 Objective

The coherent TT traffic segment should be compressed as much as possible to preserve the maximum part of the remaining bandwidth for the RC and BE traffic. This idea follows the practice from the FlexRay bus or Profinet, where the dedicated communication segment is allocated for the TT traffic. The TT traffic can be scheduled at the beginning of the integration cycle, and the remaining coherent gap in the integration cycle without the TT traffic is preserved for the RC and BE traffic. The gap, which is the shortest among all the links, is denoted as a *minimal guaranteed gap* (see Fig. 4.2). Considering the constraints and aspects above, the goal of the scheduling is to find a feasible schedule for the TT messages, which maximizes the minimal guaranteed gap.

4.4 Algorithm

The described problem is extensively complex as it involves scheduling together with routing. The algorithm that would solve the whole problem at once would put extreme demands on the computational resources or time needed to find the solution. Thus, the incremental scheduling problem proposed in the paper is decomposed into three subproblems to tackle the computational effort. The solution of each subproblem fixes some decisions for the subsequent problem. Hence, the incremental scheduling algorithm can be considered as being divided into three stages. In the first stage (Sec. 4.4.1), the routing of the messages is established. In the second stage (Sec. 4.4.2), the algorithm finds the assignment of the messages to the particular integration cycles. The transmission times for each message in each link are decided in the last stage (Sec. 4.4.3).

4.4.1 Messages routing problem

The network topology is often a tree in industrial networks. It means that there are no cycles and, therefore, only one possible path from a communication endpoint to any other endpoint exists. Thus, the determination of the routing is trivial in such a case. However, the TTEthernet does not restrict the network topology to the tree. The cycles introduce new redundant paths for messages that can serve as a backup during a partial network malfunction. Moreover, the appropriate selection of the routing path of the message can balance the load among the network. Thus, redundant paths can remove the bottleneck of the tree topology. However, the

TT messages have to know which path they are routed through in advance.

$$
\begin{aligned}
\min \quad & G \\
\text{s.t.} \quad & g_{l,m} \leq G && \forall k_{l,m} \in K \\
& \sum_i c_i^{l,m} \cdot \frac{cc}{p_i} \cdot x_{i,l,m} = g_{l,m} && \forall k_{l,m} \in K \\
& \sum_l x_{i,l,m} = 1 && \forall m_i \in M;\ \forall m \in Q_i \\
& \sum_m x_{i,l,m} \geq 1 && \forall m_i \in M;\ l = q_i \\
& \sum_l x_{i,l,m} \leq \sum_k x_{i,m,k} && \forall m_i \in M;\ \forall m \in E^R \\
& \sum_l x_{i,l,m} \leq 1 && \forall m_i \in M;\ \forall m \in E^R \\
& \sum_l x_{i,l,m} \geq \frac{\sum_k x_{i,m,k}}{deg^-(m)} && \forall m_i \in M;\ \forall m \in E^R \\
& s_{i,m} \geq 1 + s_{i,l} - B + B x_{i,l,m} && \forall m_i \in M;\ \forall k_{l,m} \in K \\
& s_{i,m} \leq 1 + s_{i,l} + B - B x_{i,l,m} && \forall m_i \in M;\ \forall k_{l,m} \in K \\
& s_{i,l} = 0 && \forall m_i \in M;\ l = q_i \\
& x_{i,l,m} = 1 && \forall m_i^{l,m} \in \widetilde{MI} \\
& x_{i,l,m} \in \{0,1\} && \forall m_i \in M;\ \forall l,m \in E \\
& g_{l,m} \in \mathbb{Z}_0^+ && \forall l,m \in E \\
& G \in \mathbb{R} \\
& s_{i,l} \in \mathbb{Z}_0^+ && \forall m_i \in M;\ \forall l \in E
\end{aligned}
\tag{4.1}
$$

Therefore, the first stage of the algorithm finds the routing. In accordance with the claim above, the algorithm aims to find such a routing that the network load is as balanced among the links as possible. The balanced network gives a good premise that the resulting communication schedules will be shorter than in the case of an unbalanced network. Thus, this routing objective corresponds to the aims of the scheduling algorithm.

An MILP model is used to solve the routing subproblem and decide routing tree R_i for each message.

The binary variable $x_{i,l,m}$ decides whether the message m_i is routed through the link $k_{l,m}$, and variable $g_{i,m}$ represents the load of the link $k_{l,m}$. The real variable F is, consequently, the load of the busiest link. The auxiliary variable $s_{i,l}$ assigns a numerical label to each node e_l for each message m_i. The label determines the depth of the node e_l in the routing tree R_i. The artificial constant B represents any number that is bigger than the maximal depth ($\max s_{i,l}$). Parameter $c_i^{l,m}$ represents the transmission time of message m_i in link $l_{l,m}$. Note, that if the links are configured to have a different bandwidth, then the transmission time of the same message varies among the links.

The objective of the MILP model minimizes the load of the busiest link. The first constraint, together with objective, ensures that the value of G equals the load of the busiest link. The second constraint calculats the load $g_{l,m}$ for each link. The third and fourth constraints force the routing path for each message to also contain the receiving nodes and the transmitting node. The fifth, sixth, and seventh constraint assure that the redistribution nodes serve as the inner nodes of the routing tree. The eighth, ninth and tenth constraints guarantee the routing tree of any message not to contain the cycle. B represents any constant that is larger than the number of nodes in the topology. Its aim is to make the model ignore the constraints if the value of $x_{i,l,m}$ is equal to zero. The eleventh constraint forces the resulting routing to satisfy the backward compatibility.

The routing tree R_i defines the set of links in which the message is to be scheduled and specifies the precedence relations among the message instances.

4.4.2 Integration cycle assignment problem

To distribute the messages among the integration cycles, we used an idea from the multiprocessor scheduling area. In the area, if all the workload of the tasks is distributed among the processors evenly, then the part of the integration cycle used by the TT communication has a good chance to be minimal. Following that, the algorithm tries to distribute the messages among the integration cycles evenly. All the precedence constraints, the time lags imposed by the switch delay τ, and the real-time constraints are relaxed here. The integration cycle assignment problem is formulated as the following MILP model 4.2.

The binary variable $a_{i,j} = 1$ iff message m_i is assigned to the integration cycle $j \in \{0 \dots p_i\}$. Similarly, the binary parameter $\tilde{a}_{i,j} = 1$ iff message m_i was scheduled to the integration cycle $j \in \{0 \dots p_i\}$ in the original schedule. The first constraint assures that the first message occurrence appears in exactly one of the possible integration cycles. Thus, it satisfies the completeness constraint. The second constraint makes the variable z to have the value equal to or greater than the time needed to exchange all the messages in any integration cycle of any link in the network. The constraint is evaluated for each link and each integration cycle in the cluster cycle so that the transmission times of all the message occurrences assigned to the particular integration cycle in the given link are summed up. The resulting total time must be less than or equal to variable z. The aim of the MILP model is to find such an assignment that minimizes z. Thus, the maximal time needed for the message exchange among all the resources is minimized. The third and fourth constraint force the messages to be assigned to the integration cycle, which can

satisfy the release date and deadline constraints. The last constraint forces the messages from the original schedule to be assigned to the corresponding integration cycle in the new schedule.

$$
\begin{aligned}
\min_{a_{i,j}} \quad & z \\
\text{s.t.} \quad & \sum_{j} a_{i,j} = 1 && \forall i \in M \\
& \sum_{m_i \in k_{l,m}} c_i^{l,m} \cdot a_{i,j \bmod p_i} \leq z && \forall j,l,m \mid j \in \{0 \dots \frac{cc}{ic}\} \\
& a_{i,j} = 0 && \forall i,j \mid d_i < j \cdot ic \\
& a_{i,j} = 0 && \forall i,j \mid r_i > (j+1) \cdot ic \\
& a_{i,j} = 1 && \forall i,j \mid \tilde{a}_{i,j} = 1 \\
& a_{i,j} \in \{0,1\};\ z \in R && \forall i,j
\end{aligned}
\tag{4.2}
$$

The resulting assignment balances the load among the integration cycles, follows the routing of the messages, and preserves the timing and backward compatibility constraints.

4.4.3 Link schedules creation problem

The constraint programming model is employed to create the resulting schedule. For the description of the model, the IBM CP Optimizer formalism [35] will be used. The CP model is based on so-called interval variables which, in our case, represent each message instance $m_i^{l,k} \subset MI$ in the schedule. The set of message instances to be scheduled on a particular link is known since the routing of the messages has been already decided. For each interval variable, the solver decides its start time. In the model, the time is considered as a relative offset to the start time of the integration cycle. Thus, two message instances that are scheduled with the same offset in the integration cycle, but with a different integration cycle are considered as being scheduled at the same time.

The objective of the scheduling is to minimize the part of the integration cycle used by the TT communication:

$$\min \max_{i,l,m} \text{endOf}(m_i^{l,m})$$

The length of the message is preserved by:

$$\text{lengthOf}(m_i^{l,m}) = c_i^{l,m} \quad \mid \forall i \in M; l,m \in E$$

Further constraints have to be introduced to satisfy the timing constraints. Due to the known message instance to the integration cycle assignment, the release

date $\hat{r}_i$ and deadline $\hat{d}_i$ relative to the integration cycle in which message m_i is transmitted are also known. Thus, the timing constraints can be defined as the start time limitation of the related interval variable:

$$\begin{aligned} \text{startMin}(m_i^{l,m}) &= \hat{r}_i \quad | \forall m_i^{l,m} \in MI \\ \text{endMax}(m_i^{l,m}) &= \hat{d}_i \quad | \forall m_i^{l,m} \in MI \end{aligned}$$

Similarly, the backward compatibility is assured to be satisfied by:

$$\text{startOf}(m_i^{l,m}) = \text{startOf}(\tilde{m}_i^{l,m}) \quad | \forall m_i^{l,m} \in \widetilde{MI}$$

where $\text{startOf}(\tilde{m}_i^{l,m})$ denotes the offset of the message instance in the original schedule.

The contention-free constraint is necessary to be satisfied next. From the model point of view it means that no two message instances, which appear in the same integration cycle and on the common link, can overlap. As the assignment of the message instances to the links (routing) and integration cycles is already decided, it can be trivially deduced in which link and integration cycle message m_i appears. Let us denote $MI_i^{l,m}$ the set of message instances which appear in the same integration cycle ic_i and link $k_{l,m}$. Now, the contention-free constraint is stated as:

$$\text{noOverlap}(MI_i^{l,m}) \quad | \forall i \in I; l, m \in K$$

where noOverlap is a CP operator that keeps all the interval variables in the given set to be scheduled in distinct time intervals.

Finally, it is necessary to keep the precedences among message instances. In this case, the precedence constraints are given by the routing of the message R_i and by the technical delay caused by switching the logic in the redistribution nodes. Let $P_i^{l,m}$ be the set of predecessors of the message instance $m_i^{l,m}$ in R_i. The message instance $m_i^{k,l}$ is part of the $P_i^{l,m}$ if and only if $x_{i,k,l} = 1$ (see the MILP model for the routing). Consequently, the precedence constraint is formulated as:

$$\text{endBeforeStart}(p, m_i^{l,m}, \tau) \quad | \forall p \in P_i^{l,m}; \forall i \in M; l, m \in K$$

With these constraints, the CP model for the message scheduling is defined completely.

4.5 Experimental results

The proposed scheduling method was tested on a PC with Intel ® Core™ i7-4610M CPU (two cores with 3 GHz and hyper-threading) and 32 GB RAM. The algorithm uses the Gurobi ILP Solver for determining the messages routing and for solving the Integration cycle assignment problem. The Link schedules creation problem was solved by the IBM CP Optimizer. The time to solve a benchmark instance was limited to 5 min.

The benchmark instances used in this study were synthetically generated. The benchmark generator defines the topology first. In this study, a random graph

topology is generally used. Consequently, the message parameters are generated randomly, considering the imposed limitations. These imposed limitations are described individually in detail for each test in the following sections. Each message is assigned to either the broadcast, multicast, or unicast group. Finally, the set of receivers is generated for each message according to the group.

The generation of the incremental scheduling benchmark instances was performed in reversed order, i.e., the benchmark instance for the last incremental iteration was generated first. Then the original instances for the incremental scheduling are made. For example, to generate an instance for the penultimate incremental iteration, the last instance is taken, and some of the messages, nodes, and links are removed from the instance according to the given pruning ratio.

To provide statistically significant results, thirty instances of each benchmark set were generated, and the presented values represent the mean value from all these instances.

4.5.1 Evaluation of the routing algorithm

The proposed algorithm uses the message routing that aims to support the scheduling objective by the uniform scattering of the messages among the links. Thus, it unloads the communication on the most utilized links. To evaluate the algorithm's performance, the proposed routing method is compared to the Shortest path tree (SPT) routing method. The SPT routing method minimizes the number of hops the message needs to take to get from the transmitter to the receivers. It optimizes the overall bandwidth utilized by the communication on the network, but it does not prevent the bottlenecks caused by the individual overutilized links.

The comparison of both methods is presented in Fig. 4.4. For the evaluation, the complete scheduling algorithm was executed while the MILP or SPT method was used for the routing.

Benchmark sets with 200 TT messages generated with periods in a range from one to three integration cycles and with Ethernet frame lengths in full range (i.e., up to 1500 bytes) were used to test the routing algorithm.

Fig. 4.4 presents how the method is able to utilize the redundant links in the graph. The x-axis denotes the number of redundant links added to the tree topology. The left y-axis, related to the Schedule SPT and Schedule ILP lines, represents the duration of the part of the integration cycle used for the TT communication and the right y-axis, related to the New link ILP and New link SPT line, represents the volume of the data exchanged through the newly introduced links.

As can be observed from "Schedule SPT" and "Schedule ILP" lines in Fig. 4.4, adding nine additional edges to the tree topology can shorten the duration of the part of the incremental cycle used by the TT communication to almost 50%. However, the benefits of the redundant links is that it can utilize the proposed method based on the MILP model (labeled as "Schedule ILP" in the figure) better than the method based on the SPT algorithm. On the tree topology, both methods behave equally as the routing tree is already decided by the topology. However, as the number of additional links is increasing, the proposed method acts significantly better than the SPT method.

The "New link" lines, on the other hand, shows how the significance of the

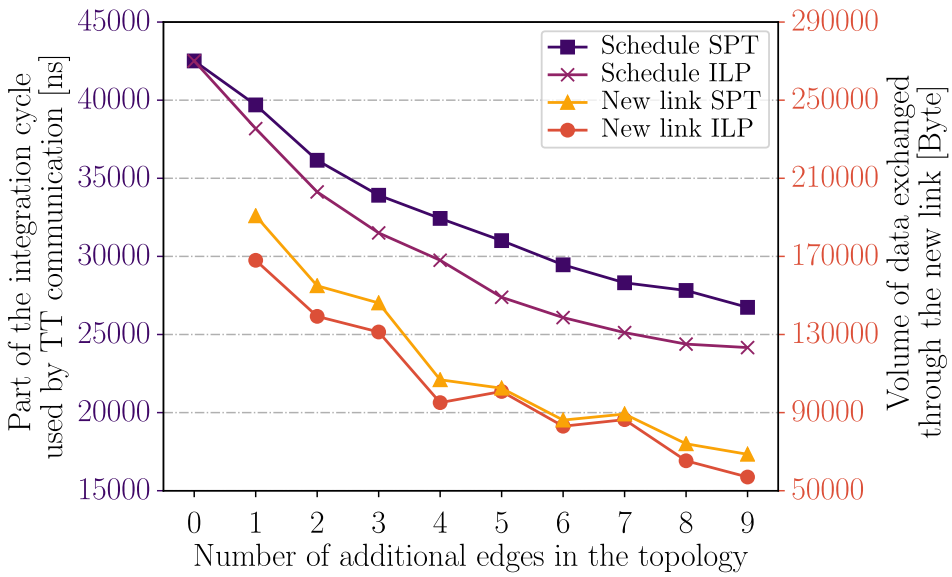

Figure 4.4: The evaluation of the routing quality

newly introduced links evolves with the number of additional edges in the topology. The routing algorithms are able to forward through the last added message only about one-third of the data volume compared to the data volume it was able to forward through the first added link.

4.5.2 Impact of the incremental scheduling on the schedule

The backward compatibility constraint introduced to the scheduling problem causes the overhead in the resulting schedule. To measure the overhead of the incremental scheduling over the non-incremental scheduling, another experiment was performed. The new set of benchmark instances was generated with ten incremental iterations. In the case of the incremental scheduling, the resulting schedule from the previous iteration was used as the original schedule. The first incremental scheduling iteration has an empty original schedule. All the messages were generated with a period in a range from one to three integration cycles and with Ethernet frame lengths in full range. The last incremental iteration instances contained 500 TT messages.

The results from the experiment are presented in Fig. 4.5. The incremental scheduling iteration is situated on the x-axis of the graph, and the schedule duration of the part of the integration cycle used by TT communication is on the y-axis. The dark purple row represents the result from the algorithm where the original schedule is considered, while the orange row represents the results of the algorithm, assuming no original schedule is given.

The result for the first incremental iteration is the same for incremental and non-incremental scheduling as no original schedule is used and, consequently, no backward compatibility constraint is applied in both cases. The most notable change

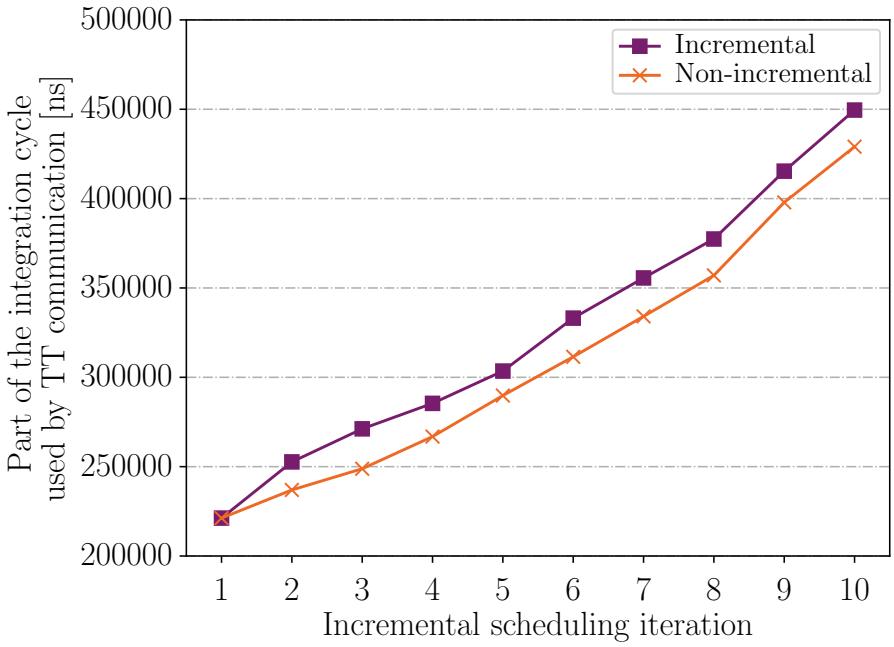

Figure 4.5: The difference between the incremental and non-incremental scheduling

in the difference between the incremental and non-incremental scheduling is present in the second incremental iteration. The prolongation caused by new messages in the case of the incremental scheduling is almost twice as much compared to the prolongation caused by the new messages in the case of the non-incremental scheduling. This causes the fact that the messages that were already scheduled in the first incremental iteration cannot be moved in the incremental scheduling. Thus, the new messages cannot be incorporated in such an efficient way as in the case of the non-incremental scheduling. However, this overhead also introduces a new porosity to the schedule. The further incremental scheduling iterations are able to use this porosity to place the new signals into those gaps efficiently. That is the reason why the overhead stays almost constant in the future scheduling iterations, and the difference between the incremental and non-incremental schedule is almost the same as can be observed in the graph.

4.5.3 Evolution of the schedule utilization over incremental iterations

To support the statement from the previous section, the way how the utilization of the schedule evolves with the incremental iterations has been studied more deeply.For the purpose of this section, the utilization of the schedule (or just *utilization*) is defined as the portion of the part of the integration cycle used for the

TT communication that is utilized by the message transmission averaged over all the integration cycles. In other words, considering only the part of the integration cycle used by the TT communication, the utilization represents the portion of the time when the link is busy. In order to test the utilization, similar benchmark instances were generated to the testing of the impact of the incremental scheduling. The only exception is the topology of the network. To avoid the impact of the routing on the porosity test, the generated instances use a tree topology. Two different measurements were performed.

Firstly, the average of the schedule utilization over the whole network (all the links) has been measured. The resulting graph can be seen in Fig. 4.6. The figure

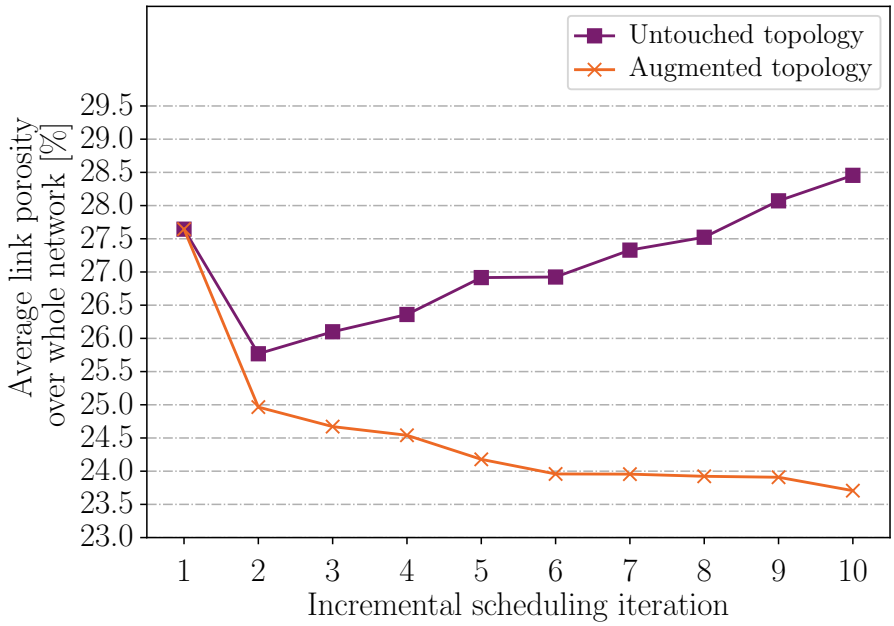

Figure 4.6: The average utilization of the communication over the whole topology

presents the evolution for two cases: the dark purple line shows how the utilization evolves when the topology is not extended during the incremental iterations; the orange line shows the utilization evolution when the topology grows together with the number of messages. The x-axis represents the incremental scheduling iteration, and the y-axis represents the average utilization of the schedule in percent. This unutilized/free space can be used by the new messages in the future incremental scheduling iteration that are local (their transmitter and receivers are close to each other according to the network topology). Note that less than 30 % of the schedule is used according to the graph. This small amount is caused by the tree topology, where the links close to the leaves of the topology are rarely used. The figure also shows that the overall schedule utilization increases if there is no topology extension which is caused by adding new messages to the schedule and the significant part

of them are local messages (thus, the part of the integration cycle used by the TT communication is not prolonged too significantly). If the topology is extended, the new sparse links introduce a lot of unutilized bandwidth to the schedule, which causes a decrease in the overall utilization.

Opposed to the new local messages that, as can be observed from Fig. 4.6, can be easily incorporated into the schedule without prolongation of the part of the integration cycle used by TT communication because of the low utilization of the links close to leaves, the new messages that need to transit the root node could cause the prolongation of the part easily. To study how the schedule utilization can impact the scheduling of these messages, the second measurement was performed where the utilization was calculated only on the most utilized link.

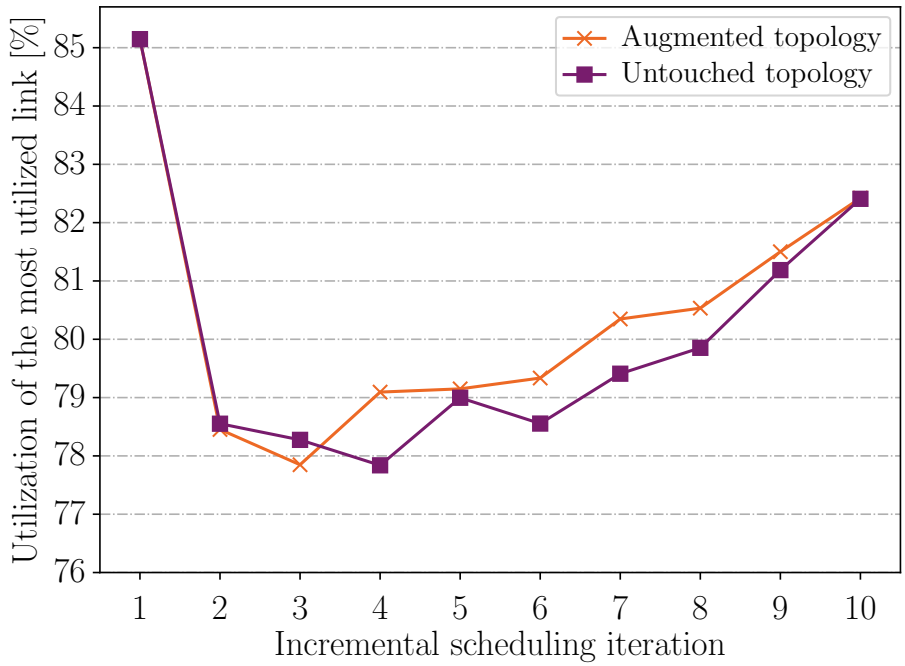

Figure 4.7: The porosity of the most utilized link

The utilization of the most utilized link is presented in Fig. 4.7. Here, the utilization is about 77 to 86 %, depending on the particular incremental scheduling iteration. This makes it harder to incorporate the long-distance messages efficiently into the schedule without prolongation of the part of the integration cycle used by the TT communication. The figure also shows that there is no significant difference for the long-distance messages, whether the topology is extended (but no cycles introduced) or not as the topology does not influence the flow of the messages over the most utilized link.

These graphs also support the statement from the last section as both the

utilization metrics are the worst in the first incremental scheduling iteration, making the second scheduling iteration the most difficult. After that, the utilization metric changes slowly and, thus, the difference between the efficiency of the incremental and non-incremental scheduling is similar.

4.5.4 Scalability of the scheduling algorithm

The previous experiments aimed to study the quality and the impact of the incremental scheduling on the resulting schedule. However, the scalability of the algorithm and its computational complexity needs to be examined next. The scalability of the algorithm is strongly dependent on three factors - the lengths of the messages, the distinct message periods, and, finally, the number of messages in the instance. The impact of all of these factors on the algorithm's runtime will be studied in the following subsections.

D.1 The scalability of the algorithm based on the message length

Firstly, the influence of the maximal allowed message length is presented. For this experiment, the instances with the maximal allowed message lengths from 1 byte to 1500 bytes of the payload were generated. To ensure that the finding of the optimal schedule and proving that the found solution is optimal will be accomplished in twenty minutes (the time limit for the computation of one benchmark instance is extended for all the scalability tests), the instances contained just 50 messages. The period of the generated messages is randomly chosen from one, two, four, or eight times the duration of the integration cycle. The results of the test are presented in Fig. 4.8.

The x-axis of the graph represents the maximal allowed message length while the y-axis represents the time needed to solve the given instance set in the logarithmic scale. The orange line presents the time needed for the routing, the red line presents the time needed for finding the assignment of the messages to the incremental cycles, the magenta line represents the time needed to find a solution and to prove that its value is at maximum 1 % off the optimum, and the dark purple line is the total time needed for the creation of the optimal schedule. Note that the error bars represent the range in which the results for the particular benchmark set were acquired (the lower cap is minimal obtained value, and the upper cap is maximal obtained value) rather than the standard deviation because the standard deviation was often larger than the mean (and it is not possible to depict them in a logarithmic scale). The position of the markers was slightly adjusted for each line of the graph, even if they represent the same value on the x-axis, to make the reading of the error bars easier.

The graph shows that the major part of the time is consumed by the link scheduling algorithm. The time complexity of the routing algorithm and also the messages to the integration cycle assignment algorithm does not depend on the message lengths at all. On the other hand, the time demands of the link scheduling algorithm are almost constant for messages with lengths up to 32 bytes, and then it grows exponentially. Also, the time range for solving the instances was much wider in the case of instances with a payload larger than 64 bytes. The variance in time demands for those instances is caused by the uncertain difficulty of proving that the current scheduling solution is optimal. It can also be read, from the figure,

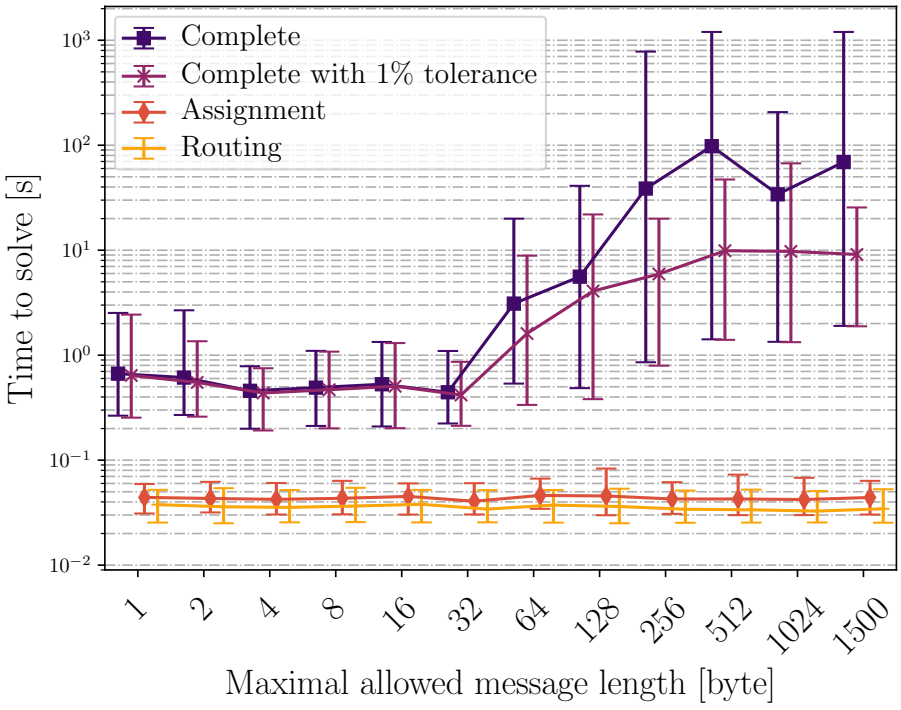

Figure 4.8: The scalability of the scheduling method according to used message lengths

that the tolerance of 1 % can reduce the scheduling time by one order in the case of messages with a full variety of their lengths.

D.2 The scalability of the algorithm based on the message periods

The second experiment evaluates the influence of the time demands on the allowed set of message periods. Two different scenarios are commonly used in practice - messages with arbitrary periods and messages with harmonic periods. The messages with arbitrary periods can have periods equal to any integer multiple of the integration cycle, while, in the case of harmonic periods, the messages can have periods only equal to any duration of the integration cycle multiplied by a power of two. The harmonic periods ensure the shortest possible cluster cycle, which can grow fast in the case of arbitrary periods because the cluster cycle is equal to the least common multiple of all the possible periods. To examine the behavior of the algorithm in both scenarios, two separate tests were performed.

Each benchmark instance contains 50 messages with up to eight bytes of payload. The general graph topology has been used. The results for the messages with the arbitrary periods are presented in Fig. 4.9, and the results for the messages with

the harmonic periods are shown in Fig. 4.10. The x-axis of the graphs represents the maximal allowed message period used.

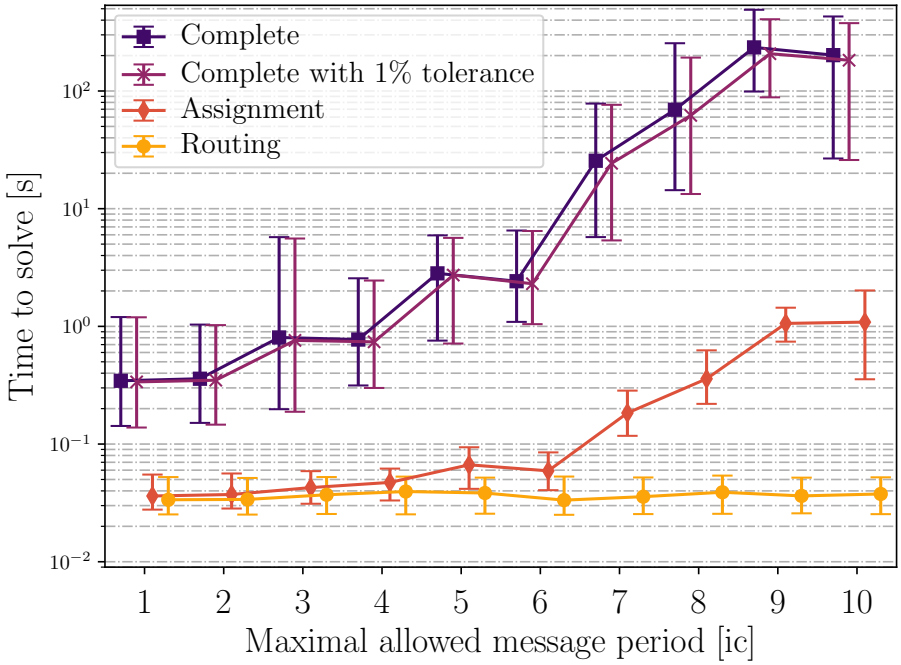

Figure 4.9: The scalability of the scheduling method according to the used message periods

The routing algorithm is not dependent on the set of used periods, and its computing demands stay low during the whole evaluation. The assignment algorithm is affected by the number of the used message periods because the messages with longer periods have more possible integration cycles to be assigned to. The computational complexity of the link scheduling algorithm grows exponentially with an increasing set of possible periods. However, the computational demands grow much steeper in the case of arbitrary periods. It is caused by the difficulty in the scheduling of the messages whose periods are co-prime. This statement is also supported by the observation that the computational complexity grows most significantly if a new prime period is introduced to the instance (specifically, if the messages with periods $5 \cdot ic$ and $7 \cdot ic$ are introduced in our case). The second reason is that the long cluster cycle (e.g., $2520 \cdot ic$ for the case with a maximal allowed period equal to $10 \cdot ic$) means a significant increase in the scheduling model variable domains for the constraint programming optimization. The computation complexity of the algorithm is considerably much lower if the periods are harmonic. In this case, the complexity growth is mainly cause by the prolongation of the cluster cycle, which is not significant as in the case of the instances with the arbitrary periods

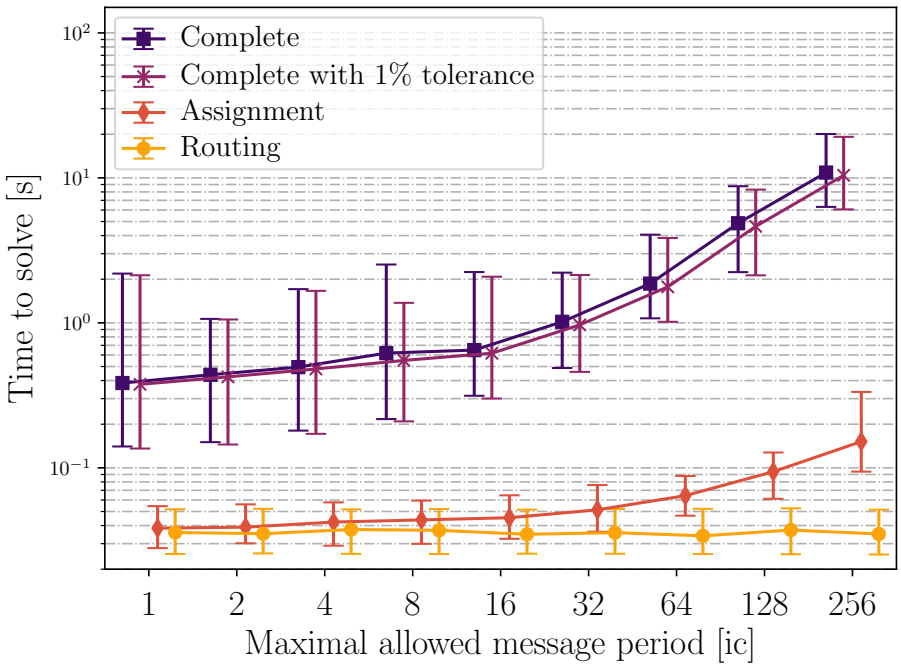

Figure 4.10: The scalability of the scheduling method according to the used harmonic periods

because the cluster cycle is only as long as the longest period here. The graphs also show that the benevolence of the loss of 1 % in the optimality does not help to reduce the time demands much here.

D.3 The scalability of the algorithm based on the message counts

The most important thing is to know how the complexity grows with the increasing number of messages, which is the subject of the third scalability experiment. The benchmark instances with messages containing up to 8 bytes of payload and with a maximal allowed period equal to $8 \cdot ic$ were generated. For these benchmark instances, harmonic periods were used, and the topology of the network corresponds to a general graph. The results of the experiment are depicted in Fig. 4.11. The x-axis represents the number of messages used in the instance.

The results show that the routing algorithm and assignment algorithm are affected by the number of messages in a similar way as the link scheduling algorithm. The narrow range of the results (with the exception of instance set with 50 messages which is affected by outliers) shows that the increase in the number of messages stably influences the computational complexity. However, the complexity grows exponentially in the number of messages. While scheduling links with instances containing 100 messages to optimality took 1 s, the instances with 2000 messages

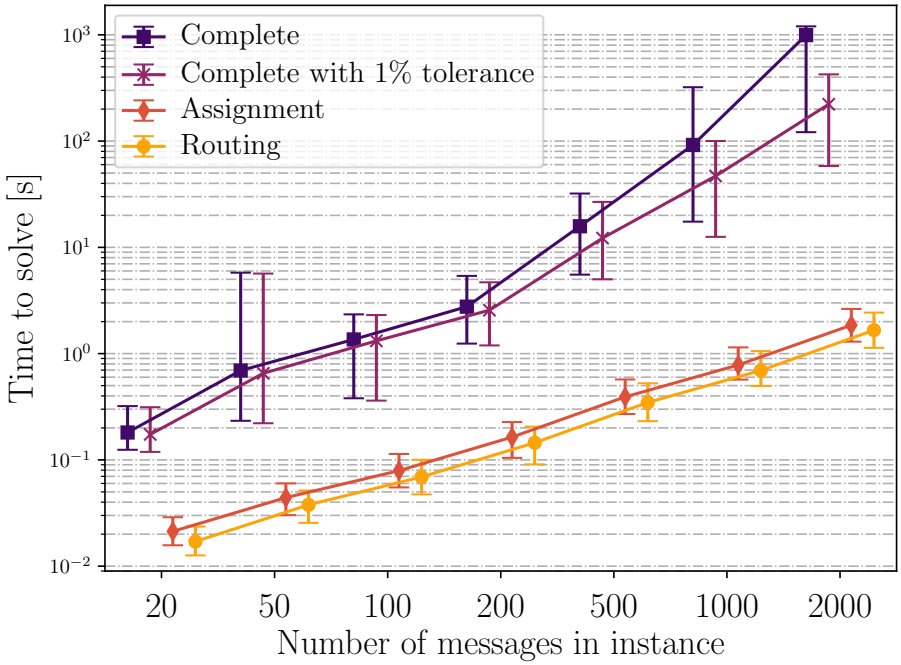

Figure 4.11: The scalability of the scheduling method according to the number of messages

needed almost 20 min. For bigger instances, the result with a proven distance of the link scheduling objective function value not further than 1 % from the optimal value can be obtained faster by an order even if this tolerance needs to be proven by the solver.

4.5.5 Evolution of the scheduling objective function in time

It is necessary to note that the optimal solution for the routing algorithm, together with the optimal solution for the link scheduling algorithm, does not ensure the optimal solution from the problem statement point of view. Thus, in practical cases, it is not needed to wait until the link scheduling algorithm proves that the current solution is optimal (or close to optimum in the case of the 1 % tolerance), and the search can be stopped sooner. This allows for creating schedules for bigger industrial size instances in a reasonable time. To show how the duration of the part of the incremental cycle used by the TT communication evolves in time, the same benchmark instance set with 2000 messages as in Section 4.5.4.3 was used. The evolution of the duration of the part of the incremental cycle used by the TT communication during the link schedule creation is presented in Fig. 4.12.

The x-axis of the figure represents the time limit for the scheduling. At the time of 1200 s, all the instances in the set were solved to optimality. The left y-axis

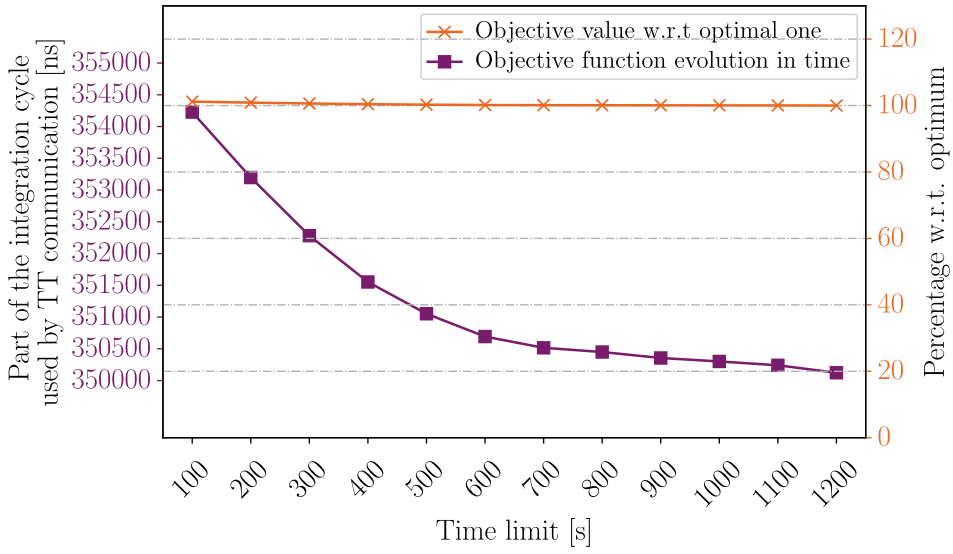

Figure 4.12: The evolution of the duration of the part of the incremental cycle used by the TT communication in the time domain

represents the average duration of the part of the integration cycle used by the TT communication over the whole benchmark set. The left y-axis represents the ratio of the obtained objective value at a particular time compared to the optimal one in percentage. Until time 100 s, the scheduling algorithm was not able to find a feasible solution for two instances out of thirty. At time 200 s, there was only one instance without finding a feasible solution. In all further times, all the instances have found a solution. The major change in the duration occurs in the beginning, and the slope is decreasing in time. Even if there is some improvement in the average duration of the part of the integration cycle used by the TT communication during the whole scheduling period, the improvement between the average duration obtained at time 100 s and 1200 s is less than 1.3 %. Thus, it is sufficient to use a solution, which is not necessarily optimal from the link schedules creation problem point of view but obtained in a shorter time, in many practical cases.

4.5.6 Scheduling of the real industrial instances

The work has been motivated by our industrial partner, who develops electronic systems for the avionic industry. This section describes the behavior of the proposed algorithm on the real instance obtained from the partner. The instance contains 1922 messages (407 unicast messages and 1515 multicast messages) with a payload of up to 1036 bytes and periods from the set {12.5 ms, 25 ms, 50 ms, 100 ms, 200 ms, 1000 ms}. The system consists of 38 nodes. The topology is based on three switches SW1, SW2 and SW3 that are mutually interconnected. Each such switch is in the topology twice called SW1_A and SW1_B, and these two instances are interconnected too. Thus, the backbone of the network is based on such a double triangle topology.

Each endpoint is connected to one of those switches. The schedule for this instance, which is optimal from all the subproblem's point of view, was obtained after less than 250 s. The schedule utilization of the most utilized link reached 96.3% while the schedule utilization averaged over the whole network was only 19.6%. This shows that the integration cycle assignment was able to distribute the messages in the most utilized link very efficiently. Figure 4.13 presents how the payload is distributed among the links in the network. The links are classified according to

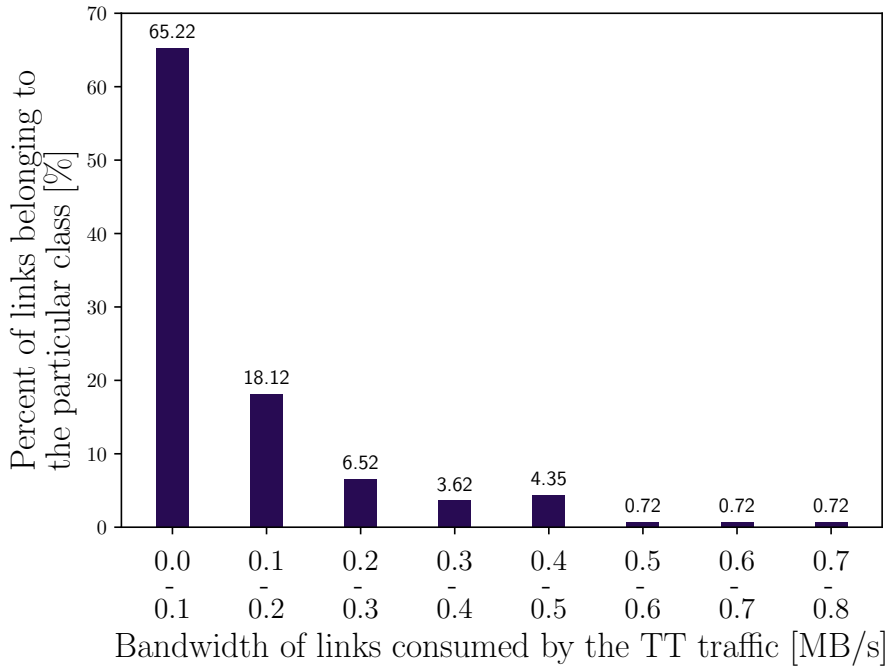

Figure 4.13: The histogram of the links utilization

the volume of the TT data transmitted through them per second. The first class includes links with the volume from 0 to 0.1 MB/s, the second one includes the links with the volume from 0.1 to 0.2 MB/s, etc. The y-axis then represents the percentage of the links that belong to the particular class. The histogram shows that almost 70 % of the links have very low utilization of the links. About 50 % of these links are empty. That means that these links are connected to the endpoints that serve as a transmitter or a receiver (note that each transmission direction of the Full-duplex physical link is represented by two separate links here). Moreover, there is just one link in the class with 0.7 - 0.8 MB/s. This link (and also the links from classes with 0.5-0.6 MB/s and 0.6-0.7 MB/s) connects the redistribution node with the communication endpoint. Thus, the routing algorithm was not able to reroute some messages from this link through another path to lighten the link. The backbone links belong to the 0.2-0.3 MB/s and 0.3-0.4 MB/s classes. The

interesting observation is that, considering the 100 MB/s TTEthernet network, the TT communication utilizes less than 1 % of the bandwidth in this instance. The rest of the bandwidth is used for the RC and BE communication.

4.6 Conclusion

The paper focuses on the problem of the incremental time-triggered communication scheduling on the TTEthernet network, and studies the influence of introduced backward compatibility on the resulting schedules.

The incorporation of the event-triggered Rate-Constrained traffic with the Time-Triggered traffic is very important for mixed-criticality applications [41]. Event-triggered communication is usually used in industrial applications nowadays. However, the pressure placed on, for example, the automotive or avionics industries to verify and certify its systems on a component and system integration level pushes system developers to use time-triggered traffic for safety-related communication as its behavior is deterministic. However, the creation of schedules for time-triggered communication is a challenging task.

We have followed the idea of separating the time-triggered traffic and event-triggered traffic already used in [21], which was inspired by the scheme of the FlexRay bus communication cycle. The objective has been to maximize the minimal guaranteed coherent gap left in each integration cycle on each link that can be continuously used by the Rate-Constrained and Best-Effort traffic while keeping the backward compatibility with the original schedules created in the previous development iterations. The problem has been decomposed into three subproblems - (i) message routing, (ii) deciding in which integration cycles each message will be exchanged and, finally, (iii) creating schedules for each link on the network. We have designed the algorithm based on the MILP formulation of the routing and integration cycle assignment problem and the CP formulation of the link scheduling problem.

The experiments show that the incremental scheduling on one side prolongs the part of the integration cycle used by the TT communication in the order of a percent (in the experiments it was about 1 %), but it brings the advantage of backward compatibility. The performance of the algorithm is dependent on the number of messages in the instance, the length, and the periodicity of the messages. However, the experiments show that the method can return good results even for industry sized instances in a few minutes.

Chapter 5

Conclusions and Future Work

This section concludes the thesis. We will discuss how the goals of the thesis were accomplished and what lessons we learned from the studied topics. The open points and directions for future work are presented in the end.

5.1 Conclusions

Demands posed on the capacity of communication systems grows together with the necessity of time-deterministic and reliable data exchange. Time-triggered communication protocols are a suitable solution for time-deterministic data exchange in critical applications. Modern time-triggered communication systems can transfer a great amount of data already, and they are able to accommodate non-time-triggered communication too. For example, FlexRay, TTEthernet, or 802.1Qbv can be considered as some of them. A common factor that determines whether the deployment of the time-triggered communication system is successful is a feasible and efficient communication schedule. However, it is a challenging task to find such a schedule as the problem belongs among the NP-hard class - i.e., there is no known algorithm that is able to solve this kind of problems in a polynomial time. In industrial-sized instances, there can be more than thousands of messages, which implies that the exact schedulers are not a suitable tool for time-trigger communication schedule creation. To overcome this inconvenience, the thesis in hand proposed a set of heuristic algorithms that can solve even industrial-sized instances in reasonable time and quality.

All the proposed algorithms aim to reach the same goal: to find such a feasible time-triggered communication schedule, which utilizes the communication system efficiently and leaves as much as possible bandwidth free for non-time-triggered communication. The problems the algorithms aims to solve are different from each other in aspects they consider and constraints derived from particular use-cases or communication systems.

The problem from Chapter 2 aims to reduce the bandwidth demands of time-triggered communication by utilizing the potential of two independent channels in FlexRay bus. The problem considers that not all the signals that are to be transmitted are necessarily fault-tolerant. Moreover, the ECUs that transmits only non-fault-tolerant messages can have only one FlexRay port implemented and, thus, can be connected to one communication channel only. The proposed scheduling algorithm employs the communication hypergraph to decide into which channel each ECU should be connected. Once the infrastructure of the network is decided, the schedule for each independent communication channel is created, such that the fault-tolerance of critical signals is preserved. According to the experimental results, the use of independent channels can save from 10 % to 45 % of communication bandwidth in real cases even without compromising the fault-tolerance of the system. Such a significant saving of the bandwidth can enable further development

or allow integration of more ECUs into one FlexRay bus and thereby save production costs.

Chapter 3 then focuses on the creation of FlexRay static segment communication schedules in the development process that is directly inspired by the automotive industry. The scheduling algorithm adapts to the requirements of the industry that demands to preserve the compatibility among resulting products. The compatibility requirement can be divided into two sub-requirements: keep compatibility among products in current development (multi-variant scheduling constraints) and keep backward compatibility with already existing products (incremental scheduling constraints). Meeting these requirements brings significant savings in the production and after-sale segment. The sophisticated algorithm has been developed to satisfy these compatibility requirements and to optimize bandwidth utilization. Moreover, the algorithm tries to preventively improve the structure of the final schedule so that it is possible to find more efficient schedules even in the further incremental scheduling iterations. The chapter also tries to solve the case when the algorithm is not able to find a feasible schedule by providing the enumerative algorithm manages network parameters. It aims to find such a network parameters configuration that allows finding the most efficient schedules. The experimental results show that the described algorithm can find efficient schedules in less than one second, even for industrial-sized instances. Additionally, to the performance and quality experiments, the influence of multi-variant and incremental scheduling constraints on the resulting schedules was studied. The schedule is linearly dependent on the portion of common signals (signals common to more product variants) and specific signals (signals specific to one particular product variant) in the instance. The incremental scheduling introduces a strong linkage between current and original schedules. It is not possible to move the signals in the schedule arbitrary as backward compatibility forces to minimize the number of changes among original and current schedule. Consequently, the free space left in the original schedule most probably does not fit to the new signals, which finally implies an allocation of extra slots. The more compact the schedule, the more likely a new signal will not fit the free space. That is the reason why the most significant increase in the number of allocated slots is in the second incremental development iteration (the first one with the original schedule), and then the increase eases. The experiments show that this phenomenon can be efficiently reduced by predictive extensibility optimization of the schedule, which helps most in the initial incremental scheduling iterations. Finally, the feasibility of the obtained results was validated directly on the FlexRay bus powered system.

Chapter 4 leaves the FlexRay bus scheduling and tackles the problem of scheduling time-triggered communication on the TTEthernet network. A similar practice, as in the case of FlexRay, was followed where the aim is to separate the segment dedicated to the time-triggered communication. The duration of the segment should be minimal possible to preserve the rest of the bandwidth for RC and BE communication classes. The backward compatibility with previous schedules is kept here to comply with the industrial needs. The creation of TTEthernet schedules is a significantly more complex task than in the case of FlexRay, mainly because the single shared medium is replaced by a set of full-duplex point-to-point connections. That means that instead of one schedule, it is necessary to create foe each physical

link of the TTEthernet network two schedules (one for each communication direction). Moreover, it is necessary to decide the path through which the message will be routed. The proposed algorithm tackles the problem complexity by dividing the whole problem into three subproblems: routing, determination of the integration cycle in which the message will be exchanged, and finally, the creation of schedules for each link. The routing subproblem and the integration cycle assignment subproblem were solved by its formulation as a MILP model. Thanks to the fixed routing and integration cycle assignment, the amount of possible combinations for the link scheduling is significantly reduced. CP model, which is employed to find time intervals in which the message is exchanged in particular links, can consequently solve a scheduling problem in a reasonable time even for industrial-sized instances. The resulting experiments show that the time demands of the algorithm are exponentially dependent on the number of messages in the instance. However, they also snow that the algorithm can find efficient schedules even in the case when the search of link schedules is stopped before finding the optimal schedule by the CP solver. The scalability is also influenced by the maximal allowed length of a message and the number of different message periods. The increase of the complexity caused by the increased number of message periods can be reduced by the use of harmonic periods instead of arbitrary ones. This can accelerate the algorithm by one order. The experiments also examined the influence of backward compatibility on the duration of the scheduled time-triggered segment and its porosity. According to the results, the backward compatibility introduces the overhead about 10 %. The backward compatibility also causes an increase of porosity in the most utilized link from 13 % to about 23 %, and the porosity keeps in the range even in further incremental scheduling iterations.

5.2 Fulfillment of the Goals

The thesis aimed to solve the stated goals. Let us list them and check if they were accomplished successfully.

1. *Examine literature, papers, and specifications related to the time-triggered communication systems and find open challenges and possible improvements in the scheduling area. Focus on the problems that obstruct introducing time-triggered communication in the industrial sector.*

   The study of the literature revealed that the commonly used time-triggered communication protocols are FlexRay and TTEthernet. Aside from these, there are also other protocols as TTP or TT CAN. However, they are not used in practice so frequently. Currently, there is also an extension for Ethernet introduced by TSN called Enhancements for Scheduled Traffic (802.1Qbv) that allows exchanging time-triggered communication over the Ethernet network. This standard, similar to TTEthernet, seems to be a common choice for the future. However, this standard is not examined in this thesis as the thesis started before the standard was stated. Thus, it is considered as future work here. The literature also revealed the lack of work that would be focused on scheduling in the multi-variant product development process that is often also iterative and needs to preserve the backward compatibility. Thus, this

theme affects almost the whole thesis. The literature and papers related to the individual open points that were chosen to be solved are present in the Related works section of the particular chapter. Specifically, the important information from specifications are presented in Sec. 2.1.2 and Sec. 4.2.1 and the related papers are briefly described in Sec. 2.1.3, Sec. 3.1.2 and Sec. 4.2.2.

2. *Deduce a formal description of the discovered problems. Formulate the problem constraints and objectives while considering the safety-relevant and time-deterministic behavior.*

   The problem statements were deduced from the protocol specifications and related papers. Each chapter of the thesis presents one problem statement that the chapter aims to solve. The FlexRay static segment scheduling problem with independent communication channels is examined in Sec. 2.2, while the multi-variant scheduling problem that preserves backward compatibility is described in Sec. 3.2. Sec. 4.3 states formally the problem of creation of backward compatibility compliant schedules for the time-triggered communication in TTEthernet network.

3. *Develop exact or heuristic algorithms that can solve the scheduling problems for industrial-sized instances. The schedules have to be obtained in a reasonable time for the product development process.*

   Section 2.3 describes the heuristic method for solving problem from Sec. 2.2 - FlexRay static segment scheduling with two independent channels. The method is based on a two-step iterative algorithm where, in the first step (described into detail in Sec. 2.3.1), ECUs to channel assignment is decided. The assignment also determines in which channel or channels will be each message transmitted and if the message needs to transit the gateway ECU. In the second step (described in Sec 2.3.2), schedules for both channels are created. The balanced assignment from the first step does not ensure the ideal starting point for scheduling. Thus, both steps are called iteratively while the parameter setting of ECU to channel assignment algorithm is adapted with respect to the results from the scheduling step and the best-found schedule is provided at the end.
   The problem from Sec. 3.2 - creating time-triggered communication schedules for multiple variants of FlexRay powered system while considering the backward compatibility with original schedules - is solved by the heuristic method described in Sec. 3.3. The method reads the original schedules first and checks if these schedules are feasible even for a new set of variants. If some scheduling conflicts are found (i.e., backward compatibility has to be violated to obtain a feasible schedule), these conflicts are solved by removing conflicting messages from the schedule. Conflict solving algorithm aims to remove the smallest possible number of messages to limit backward compatibility violation as much as possible. Removed messages are merged with the set of new messages and introduced to the schedule in the next step. Consequently, the created schedules are optimized for future scheduling iterations. Finally, the resulting schedule is made up from separated schedules for individual ECUs by graph coloring like method.

The problem of creating schedules for the time-triggered communication on TTEthernet described in Sec. 4.3 is solved by three-stage heuristic algorithm provided in Sec. 4.4. The routing for each message through the TTEthernet network is found by ILP model, presented in Sec. 4.4.1, in the first stage. The second stage of the algorithm (Sec. 4.4.2) decides in which integration cycle will be the message scheduled. This stage significantly reduces the time complexity of the third stage. Section 4.4.3 presents the CP model for the third stage of the algorithm, which finally finds the time-triggered communication schedules for each link of the TTEthernet network.

4. *Verify the developed algorithms by experiments. Discuss the obtained results from the quality point of view as well as from the time complexity and scalability point of view. Investigate the uncommon properties of the stated problems and the proposed algorithms.*

   Each topic of the work was experimentally tested, and the results of the tests are presented and discussed in sections 2.4, 3.4 and 4.5. The experiments that aim to determine the quality of the created schedules can be found in Table 2.2 and sections 3.4.1 and 4.5.2. On the other hand, the time complexity of the described algorithms for creating FlexRay static segment schedules is presented in Table 2.3 and Section 3.4.5. In the case of scheduling algorithm for time-trigged communication on TTEthernet, the time complexity is studied more into detail because it is significantly more time demanding operation than scheduling communication on FlexRay. Thus, the scalability of the algorithm were tested from different aspects: message lengths (Sec. 4.5.4), message periods (Sec. 4.5.4) and the number of messages (Sec. 4.5.4). The evolution of the schedule quality in time is consequently presented in Sec. 4.5.5.

   Moreover, each topic of the thesis presents the experiments that are closely related to the topic itself. Figure 2.6 shows how the portion of fault-tolerant signals and portion of commons ECUs influences the number of the FlexRay static slots allocated by our scheduling algorithm for FlexRay system with two independent channels. A question of how the portion of the specific signals and portion of the common signals influences the multi-variant scheduling was answered in Sec. 3.4.2 and Sec. 3.4.3 for FlexRay static segment scheduling. The influence of incremental scheduling and introduced backward compatibility was studied and discussed in Sec. 3.4.3 and Sec. 3.4.4 for FlexRay scheduling and in Sec. 4.5.2 for TTEthernet scheduling. Moreover, the porosity introduced into the schedule by the incremental scheduling was examined in Sec. 4.5.3. Finally, the Sec. 4.5.1 evaluates the suitability of used routing algorithm for TTEthernet scheduling, and Sec. 3.4.4 shows how extensibility optimization for FlexRay scheduling can improve the schedules in upcoming scheduling iterations if the backward compatibility is used.

Bibliography

[1] H. Arabnejad and J. Barbosa. List scheduling algorithm for heterogeneous systems by an optimistic cost table. *Parallel and Distributed Systems, IEEE Transactions on*, 25(3):682–694, March 2014.

[2] ARINC (Aeronautical Radio, Inc.). ARINC 664P7: Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network. Technical report, ARINC (Aeronautical Radio, Inc.), 2009.

[3] Association for Standardisation of Automation and Measuring Systems. ASAM MCD-2 NET standard (FIBEX), 2014.

[4] AUTOSAR Development Partnership. AUTOSAR requirements on FlexRay v4.0.1, Oct 2013.

[5] AUTOSAR Development Partnership. AUTOSAR specification of FlexRay interface v3.5.0, Oct 2013.

[6] L. L. Bello. Novel trends in automotive networks: A perspective on Ethernet and the IEEE Audio Video Bridging. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–8, Sept 2014.

[7] H. Bley and C. Zenner. Variant-oriented assembly planning. *CIRP Ann. Manufacturing Technology*, 55(1):23 – 28, 2006.

[8] R. Bouhouch, H. Jaouani, W. Najjar, and S. Hasnaoui. FlexRay static section scheduling using full model. In *2nd International Conference on Advanced Communications and Computation*, pages 74–80, Oct 2012.

[9] M. Boyer, H. Daigmorte, N. Navet, and J. Migge. Performance impact of the interactions between time-triggered and rate-constrained transmissions in ttethernet. In *8th European Congress on Embedded Real Time Software and Syst.*, pages 159–168, Toulouse, France, 2016.

[10] C. Braun, L. Havet, and N. Navet. Netcarbench: A benchmark for techniques and tools used in the design of automotive communication systems. In *7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems (FeT), Toulouse, France*, Nov 2007.

[11] A. Buiga. Investigating the role of MQB platform in Volkswagen Group's strategy and automobile industry. *Int. J. Academic Research in Business and Social Sciences*, 2(9):391–399, September 2012.

[12] R. Carvajal, J. Aguero, B. Godoy, and G. Goodwin. EM-based maximum-likelihood channel estimation in multicarrier systems with phase distortion. *Vehicular Technology, IEEE Transactions on*, 62(1):152–160, Jan 2013.

[13] Y. Chen, S. Sanghavi, and H. Xu. Improved graph clustering. *Information Theory, IEEE Transactions on*, 60(10):6440–6455, Oct 2014.

[14] S. S. Craciunas and R. S. Oliver. Combined task- and network-level scheduling for distributed time-triggered systems. *Real-Time Syst.*, 52(2):161–200, Mar 2016.

[15] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner. Scheduling real-time communication in IEEE 802.1Qbv Time Sensitive Networks. In *Proc. 24th Int. Conf. Real-Time Networks and Systems*, pages 183–192, Brest, France, 2016.

[16] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner. Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks. In *Proc. 24th Int. Conf. on Real-Time Networks and Syst. (RTNS)*, pages 183–192, Brest, France, 2016.

[17] S. S. Craciunas, R. S. Oliver, and V. Ecker. Optimal static scheduling of real-time tasks on distributed time-triggered networked systems. In *Proc. IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–8, Barcelona, Spain, 2014.

[18] P. F. d. Souto, P. Portugal, and F. Vasques. Reliability evaluation of broadcast protocols for FlexRay. *IEEE Trans. on Veh. Technol.*, 65(2):525–541, Feb 2016.

[19] J. Dvořák and Z. Hanzálek. Multi-variant time constrained FlexRay static segment scheduling. In *10th IEEE Workshop on Factory Communication Systems (WFCS)*, pages 1–8, Toulouse, France, 2014.

[20] J. Dvořák and Z. Hanzálek. Incremental Multi-Variant Flexray Static Segment Scheduler, 2018.

[21] J. Dvořák, M. Heller, and Z. Hanzálek. Makespan minimization of Time-Triggered traffic on a TTEthernet network. In *Proc. 13th IEEE Int. Workshop on Factory Commun. Sys. (WFCS)*, pages 1–10, Trondheim, Norway, 2017.

[22] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Zomaya, S. Foufou, and A. Bouras. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *Emerging Topics in Computing, IEEE Transactions on*, 2(3):267–279, Sept 2014.

[23] W. Fenske, S. Schulze, D. Meyer, and G. Saake. When code smells twice as much: Metric-based detection of variability-aware code smells. In *IEEE 15th Int. Work. Conf. Source Code Analysis and Manipulation (SCAM)*, pages 171–180, Bremen, Germany, 2015.

[24] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176 – 190, 2008.

[25] G. Gutin, T. Jensen, and A. Yeo. Batched bin packing. *Discrete Optimization*, 2(1):71 – 82, 2005.

[26] Z. Hanzalek, D. Benes, and D. Waraus. Time constrained FlexRay static segment scheduling. In *10th International Workshop on Real-Time Networks (RTN), In conjunction with Euromicro Conference on Real-Time Systems (ECRTS), Porto, Portugal*, July 2011.

[27] Z. Hanzalek and T. Pacha. Use of the fieldbus systems in academic setting. In *Proceedings on real-time systems education III*, pages 93–97, 1999.

[28] Y. Hua, X. Liu, W. He, and D. Feng. Design and implementation of holistic scheduling and efficient storage for FlexRay. *Parallel and Distributed Systems, IEEE Transactions on*, 25(10):2529–2539, Oct 2014.

[29] International Organization for Standardization. ISO 17458 - FlexRay communications system, 2015.

[30] Z. Ivković and E. L. Lloyd. *Fully Dynamic Bin Packing*. Netherlands, Springer, 2009.

[31] Ixia. Automotive ethernet: An overview, White paper, 2014.

[32] M. Kang, K. Park, and M.-K. Jeong. Frame packing for minimizing the bandwidth consumption of the FlexRay static segment. *Industrial Electronics, IEEE Transactions on*, 60(9):4001–4008, Sept 2013.

[33] J. H. Kim, S. H. Seo, N. T. Hai, B. M. Cheon, Y. S. Lee, and J. W. Jeon. Gateway framework for in-vehicle networks based on CAN, FlexRay, and Ethernet. *IEEE Trans. on Veh. Technol.*, 64(10):4472–4486, Oct 2015.

[34] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer. The time-triggered ethernet (TTE) design. In *8h IEEE Int. Symp. Object-Oriented Real-Time Distributed Computing (ISORC)*, pages 22–33. IEEE, 2005.

[35] P. Laborie. A (not so short) introduction to cp optimizer for scheduling. In *27th Int. Conf. Automated Planning and Scheduling (ICAPS)*, 2017.

[36] R. Lange, F. Vasques, R. S. de Oliveira, and P. Portugal. A scheme for slot allocation of the FlexRay static segment based on response time analysis. *Computer Communications*, 63(C):65–76, June 2015.

[37] Y. Lee, J. Kim, and J. Jeon. FlexRay and Ethernet AVB synchronization for high QoS automotive gateway. *IEEE Trans. on Veh. Technol.*, in preprint, 2016.

[38] Z. Li, H. Wan, Z. Pang, Q. Chen, Y. Deng, X. Zhao, Y. Gao, X. Song, and M. Gu. An enhanced reconfiguration for deterministic transmission in time-triggered networks. *IEEE/ACM Transactions on Networking*, 27(3):1124–1137, 2019.

[39] M. Lukasiewycz, M. Glaß, J. Teich, and P. Milbredt. FlexRay schedule optimization of the static segment. In *IEEE/ACM 7th Int. Conf. Hardware/Software Codesign and System Synthesis*, pages 363–372, Grenoble, France, 2009.

[40] Nissan Motor Corporation, Ltd. Direct Adaptive Steering ™, 2016.

[41] A. Novak, P. Sucha, and Z. Hanzalek. Efficient algorithm for jitter minimization in time-triggered periodic mixed-criticality message scheduling problem. In *Proc. 24th Int. Conf. on Real-Time Networks and Syst. (RTNS)*, pages 23–31, Brest, France, 2016.

[42] L. Ouedraogo and R. Kumar. Computation of the precise worst-case response time of FlexRay dynamic messages. *Automation Science and Engineering, IEEE Transactions on*, 11(2):537–548, April 2014.

[43] J. P. Pedroso and M. Kubo. Heuristics and exact methods for number partitioning. *European Journal of Operational Research*, 202(1):73 – 81, 2010.

[44] P. Pop, P. Eles, Z. Peng, and T. Pop. Scheduling and mapping in an incremental design methodology for distributed real-time embedded systems. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 12(8):793–811, Aug 2004.

[45] F. Pozo, G. Rodriguez-Navas, and H. Hansson. Methods for large-scale time-triggered network scheduling. *Electronics*, 8(7), 2019.

[46] O. Rottenstreich, M. Di Francesco, and Y. Revah. Perfectly periodic scheduling of collective data streams. *IEEE/ACM Transactions on Networking*, 25(3):1332–1346, 2017.

[47] SAE International. AS6802: Time-Triggered Ethernet. Technical report, SAE International, 2011.

[48] Saelig Company, Inc. EMC Pre-compliance, Testing on budget, 2016.

[49] F. Sagstetter, P. Waszecki, S. Steinhorst, M. Lukasiewycz, and S. Chakraborty. Multischedule synthesis for variant management in automotive time-triggered systems. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 35(4):637–650, 2016.

[50] E. G. Schmidt and K. Schmidt. Message scheduling for the FlexRay protocol: The dynamic segment. *IEEE Trans. Veh. Technol.*, 58(5):2160–2169, Jun 2009.

[51] K. Schmidt and E. G. Schmidt. Message scheduling for the FlexRay protocol: The static segment. *IEEE Trans. Veh. Technol.*, 58(5):2170–2179, Jun 2009.

[52] A. Sharifnassab and S. J. Golestani. On the possibility of network scheduling with polynomial complexity and delay. *IEEE/ACM Transactions on Networking*, 25(6):3850–3862, 2017.

[53] Y. H. Sheu and C. M. Ku. The intelligent FlexRay safety monitoring platform based on the automotive hybrid topology network. In *8th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, pages 289–292, July 2012.

[54] D. Singh, M. Singh, and T. Singh. A hybrid heuristic algorithm for the euclidean traveling salesman problem. In *International Conference on Computing, Communication Automation (ICCCA)*, pages 773–778, May 2015.

[55] M. Sojka, P. Pisa, D. Faggioli, T. Cucinotta, F. Checconi, Z. Hanzalek, and G. Lipari. Modular software architecture for flexible reservation mechanisms on heterogeneous resources. *Journal of Systems Architecture*, 57(4):366 – 382, 2011.

[56] T. Steinbach, F. Korf, and T. C. Schmidt. Comparing time-triggered Ethernet with FlexRay: An evaluation of competing approaches to real-time for in-vehicle networks. In *Proc. 8th IEEE Int. Workshop on Factory Commun. Syst.*, pages 199–202, Nancy, France, 2010.

[57] W. Steiner. An Evaluation of SMT-Based Schedule Synthesis for Time-Triggered Multi-hop Networks. In *31st IEEE Real-Time Syst. Symp. (RTSS)*, pages 375–384, San Diego, CA, USA, 2010.

[58] W. Steiner. Synthesis of static communication schedules for mixed-criticality systems. In *14th IEEE Int. Symp. Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW)*, pages 11–18, Newport Beach, CA, USA, 2011.

[59] W. Steiner, M. Gutiérrez, Z. Matyas, F. Pozo, and G. Rodriguez-Navas. Current techniques, trends and new horizons in avionics networks configuration. In *34th IEEE/AIAA Digital Avionics Syst. Conf. (DASC)*, pages 1–26, Prague, Czech Republic, 2015.

[60] D. Tamas-Selicean, P. Pop, and W. Steiner. Design optimization of TTEthernet-based distributed real-time systems. *Real-Time Syst.*, 51(1):1–35, 2015.

[61] D. Tamas-Selicean, P. Pop, and W. Steiner. Design optimization of TTEthernet-based distributed real-time systems. *Real-Time Syst.*, 51(1):1–35, Jan 2015.

[62] D. Tamas-Selicean, P. Pop, and W. Steiner. Timing Analysis of Rate Constrained Traffic for the TTEthernet Communication Protocol. In *18th IEEE Int. Symp. on Real-Time Distributed Computing*, pages 119–126, Auckland, New Zealand, 2015.

[63] B. Tanasa, U. Bordoloi, P. Eles, and Z. Peng. Scheduling for fault-tolerant communication on the static segment of FlexRay. In *31st IEEE Real-Time Systems Symp. (RTSS)*, pages 385–394, 2010.

[64] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin. Intra-vehicle networks: A review. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):534–545, 2015.

[65] Vector Informatik GmbH. ECU Development & Test with CANoe, 2016.

[66] Volkswagen Group. New group strategy adopted: Volkswagen Group to become a world-leading provider of sustainable mobility, 2016.

[67] R. Wallis, J. Stjepandic, S. Rulhoff, F. Stromberger, and J. Deuse. Intelligent utilization of digital manufacturing data in modern product emergence processes. *IOS Press - Advances in Transdisciplinary Eng.*, 1(1):261–270, 2014.

[68] J. Wang, P. Ding, Y. Wang, and G. Yan. Back-to-Back Optimization of Schedules for Time-Triggered Ethernet. In *37th Chinese Control Conference (CCC)*, pages 6398–6403, Wuhan, China, July 2018.

[69] B. Westfechtel. Realizing a conceptual framework to integrate model-driven engineering, software product line engineering, and software configuration management. In *3rd Int. Conf. Model-Driven Eng. and Software Development (MODELSWARD)*, pages 21 – 44, Angers, France, 2015.

[70] J. Whyte, A. Stasis, and C. Lindkvist. Managing change in the delivery of complex projects: Configuration management, asset information and 'big data'. *Int. J. Project Management*, 34(2):339 – 351, 2016.

[71] L. Zhao, P. Pop, and S. S. Craciunas. Worst-case latency analysis for ieee 802.1qbv time sensitive networks using network calculus. *IEEE Access*, 6:41803–41815, 2018.

[72] R. Zhao, G. Qin, Y. Lyu, and J. Yan. Security-aware scheduling for ttethernet-based real-time automotive systems. *IEEE Access*, 7:85971–85984, 2019.

Nomenclature

Abbreviations

AUTOSAR	AUTomotive Open System ARchitecture
BE	Best-Effort
CAH	ECU-to-Channel Assignment Heuristic
CAN	Controller Area Network
CLP	Constraint Logic Programming
CP	Constraint Programming
ECU	Electronic Control Unit
EEM	ECU Mutual Exclusion Matrix
ET	Event-Triggered
ILP	Integer Linear Programming
ISO	International Organization for Standardization
MILP	Mixed Integer Linear Programming
MIS	Maximal Independent Set
MS	Multischedule
MWIS	Maximal Weighted Independent Set problem
NP	Nondeterministic Polynomial time
RC	Rate-Constrained
SEM	Signal Mutual Exclusion Matrix
SMT	Satisfiability Modulo Theories
SPT	Shortest Path Tree
TDMA	Time Division Multiple Access
TT	Time-Triggered

Symbols

$a_{i,j}$	Decision variable for the integration cycle assignemnt problem
$\tilde{a}_{i,j}$	Predicate denoting if message m_i was scheduled to integration cycle ic_j in the original schedule

α	Gateway throughput penalization medifier
β	Channel balancing modifier
b_i	Channel of signal transmission
b'_i	b_i for image of signal i
cc	Cluster cycle
CG	Conflict graph
c_i	Payload length of signal or message i
$c_i^{l,m}$	Transmission time of message m_i in link $k_{l,m}$
D	Set of dummy signals
d_i	Deadline of signal or message i
E	Set of ECUs
E^C	Communication endpoints
E_{CG}	Edges from conflict graph
e_i	Node i
E^R	Redistribution nodes
f_i	Fault-tolerance of signal i
G	Load of the most bussy link
$g_{i,m}$	Load of the link $k_{l,m}$
G_{SLOT}	Graph for Slot scheduling
H	Number of cycles in the hyperperiod
h_i	Index of ECU transmitting slot l_i
I	Set of integration cycles
i	Index
ic	Integration cycle
j	Index
K	Set of links
$k_{i,j}$	Link between nodes e_i and e_j
L	Duration of one communication cycle
l_i	Slot with index i

M	Set of messages
m_i	Message i
MI	Set of message instances
$MI_i^{l,m}$	Set of message instances which appear in integration cycle ic_i and link $k_{l,m}$
$m_i^{l,m}$	Instance of message m_i on link $k_{l,m}$
$\widetilde{MI}$	Set of message instances present in the original schedule
MS^{ECU}	Unit multischedule
$MS_{i,j}$	Multiframe in cycle i and slot j
MS^O	Original multischedule
$\mathcal{N}$	Set of ECUs
N	Set of one port ECUs
N_{CG}	Nodes from conflict graph
N^{Comm}	Set of common ECUs
N^{GW}	Gateway ECU
NL	Ordered list of nodes
o_i	Offset in the frame of signal s_i
$\tilde{o_i}$	o_i from original assignment
o_i'	o_i for image of signal i
p_i	Period of signal/message i
Q_i	Set of receivers signal or message i
q_i	Transmitter of signal/message i
R_i	Routing tree of message m_i
r_i	Release date of signal or message i
R_i^q	Routing path of message m_i to receiver e_q
S	Set of signals
SA	Schedule for channel A
SAB	Schedule common to both channels
SB	Schedule for channel B
s_i	Signal i

$s_{i,l}$	Numerical label of node e_l for message m_i
s'_i	Image of signal i
SL	Ordered list of signals
SL^N	Set of new signals
S_{MIS}	Set of signals from the Maximal Independent Set
$\tilde{S}$	Subset of signals used in the original multischedule
S'	Set of signal images
τ	Hop delay
t_i	Slot in which signal i is transmitted
$\tilde{t_i}$	t_i from original assignment
t'_i	t_i for image of signal i
U	Number of free bits in slot
$V_{i,j}$	Binary matrix of the signal-to-variant assignment
W	Maximal frame payload length
w_i	Weight of node i in MWIS
$x_{i,l,m}$	Decision variable for the routing
y_i	Cycle in which signal i is transmitted
$\tilde{y_i}$	y_i from original assignment
y'_i	y_i for image of signal i
z	Maximal bandwidth utilization among the integration cycles

Chapter A

Curriculum Vitae

Jan Dvořák was born in Mladá Boleslav, Czech Republic, in 1989. He received his bachelor degree in electrical engineering and informatics in Faculty of Electrical Engineering (FEE) in Czech Technical Univerzity in Prague (CTU) in 2011 with bachelor thesis focused on recognition of Parkinson disease from speech records (based on digital signal processing methods). Next, he received the master of science degree in open informatics in FEE in CTU in 2013, when he had defended his thesis focused on development of massively parallel algorithm on GPU for Nurse Rerostering Problem. In the same year, Jan started his Ph.D. studies on Scheduling algorithms for time-triggered communication protocols in Department of Control engineering, FEE, CTU.

The results of his study were published in two impacted international journals - IEEE Transactions on Industrial Informatics and IEEE Transactions on Vehicular Technology. The third journal paper is currently under review in IEEE/ACM Transactions on Networking. Moreover, the results from his diploma theses were publised in Computers & Operational Research journal. His current research interests includes combinatorial optimization, scheduling, communication systems and automotive embedded systems.

Jan Dvořák
Prague, June 2020

Chapter B

List of Author's Publications

List of publications related to the thesis is included in this appendix.

Publications in Journals with Impact Factor

Jan Dvořák and Zdeněk Hanzálek. Using Two Independent Channels With Gateway for FlexRay Static Segment Scheduling. IEEE Transactions on Industrial Informatics, 12: 1887-1895, 2016. ISSN 1941-0050. **Coauthorship 50%, indexed in Web of Science, 15 citations in Google Scholar**.

Jan Dvořák and Zdeněk Hanzálek. Multi-Variant Scheduling of Critical Time-Triggered Communication in Incremental Development Process: Application to FlexRay. IEEE Transactions on Vehicular Technology, 68: 155-169, 2019. ISSN 1939-9359. **Coauthorship 50%, indexed in Web of Science, 1 citation in Google Scholar**.

Jan Dvořák and Zdeněk Hanzálek. Incremental scheduling of the Time-Triggered traffic on TTEthernet network. IEEE/ACM Transactions on Networking, In review process. **Coauthorship 50%**.

International Conferences and Workshops

Jan Dvořák and Zdeněk Hanzálek. Multi-variant time constrained FlexRay static segment scheduling. 2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014), pages: 8, May 2014. **Coauthorship 50%, indexed in Web of Science, 9 citations in Google Scholar**.

Jan Dvořák and Zdeněk Hanzálek. FlexRay static segment scheduling on two independent channels with gateway. 2015 11th IEEE World Conference on Factory Communication Systems (WFCS 2015), pages: 4, May 2015. **Coauthorship 50%, indexed in Web of Science, 2 citations in Google Scholar**.

Jan Dvořák, Martin Heller and Zdeněk Hanzálek. Makespan minimization of Time-Triggered traffic on a TTEthernet network. 2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS 2017), pages: 10, May 2017. **Coauthorship 50%, indexed in Web of Science, 7 citations in Google Scholar**.

Publications not Related to this Thesis

Zdeněk Bäumelt, Jan Dvořák, Přemysl Šůcha and Zdeněk Hanzálek. A novel approach for nurse rerostering based on a parallel algorithm. European Journal of Operational Research, 251: 624 – 639, 2016. ISSN 0377-2217. **Coauthorship 25%, indexed in Web of Science, 17 citations in Google Scholar**.

Zdeněk Bäumelt, Jan Dvořák, Přemysl Šůcha and Zdeněk Hanzálek. An acceleration of the algorithm for the nurse rerostering problem on a graphics processing unit. Lecture Notes in Management Science - 5th International Conference on Applied Operational Research (ICAOR 2013), pages: 10, 2013. **Coauthorship 25%, indexed in Web of Science, 3 citations in Google Scholar**.

Jan Dvořák
Prague, June 2020

Contributions of the thesis

The thesis in hand is focused on scheduling TT communication in two communication systems - the FlexRay bus and the TTEthernet network while considering industry-relevant constraints. For both of them, the work proposes heuristic scheduling algorithms that find feasible and close to optimal schedules within a reasonable time.

The main contributions of the thesis in the field of scheduling TT communication on FlexRay bus are:

- We provided an idea of utilizing both independent FlexRay communication channels
- We provided an idea of considering the practical product development process consisting of multi-variant management and incremental development
- We proved that the scheduling problem is NP-Hard
- We designed heuristic scheduling algorithms that solve the problems
- We examined and discussed the impact of multi-variant and incremental essence on scheduling
- We evaluated the proposed algorithms on the sets of both synthetic and real-case inspired instances
- We verified the resulting schedules on the FlexRay powered system

The main contributions in the field of scheduling TT communication on TTEthernet network are:

- We provided an idea of considering backward compatibility in the TTEthernet scheduling process
- We aimed to make part used for TT communication the most compact
- We formulated the incremental TT communication scheduling problem on TTEthernet formally
- We designed the three-stage heuristic scheduling algorithm to solve the problem
- We examined and discussed the impact of incremental essence on the TTEthernet scheduling
- We evaluation of the proposed algorithm from the quality and the scalability point of view

CTU

CZECH TECHNICAL UNIVERSITY IN PRAGUE