Design Space Exploration for Embedded Systems in Automotives

Prachi Joshi

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Haibo Zeng, Chair
Michael S. Hsiao
Sekharipuram S. Ravi
Sandeep K. Shukla
Anil Kumar S. Vullikanti
Yaling Yang

March 21, 2018
Blacksburg, Virginia

Keywords: design space exploration, automotive design, real time system design, optimization

Design Space Exploration for Embedded Systems in Automotives

Prachi Joshi

(ABSTRACT)

With ever increasing contents (safety, driver assistance, infotainment, etc.) in today's automotive systems that rely on electronics and software, the supporting architecture is integrated by a complex set of heterogeneous data networks. A modern automobile contains up to 100 ECUs and several heterogeneous communication buses (such as CAN, FlexRay, etc.), exchanging thousands of signals. The automotive Original Equipment Manufacturers (OEMs) and suppliers face a number of challenges such as reliability, safety and cost to incorporate the growing functionalities in vehicles. One of the important challenges in automotive design is the efficient and reliable transmission of signals over communication networks such as CAN and CAN-FD. With the growing features in automotives, the OEMs already face the challenge of saturation of bus bandwidth hindering the reliability of communication and the inclusion of additional features. In this dissertation, we study the problem of optimization of bandwidth utilization (BU) over CAN-FD networks. By carefully optimizing signal-to-frame packing, the CAN-FD BU can be reduced. In Chapter 3, we propose a method for offset assignment to signals and show its importance in improving BU. One of our contributions for an industrial setting is a modest improvement in BU of about 2.3%. Even with this modest improvement, the architecture's lifetime could potentially be extended by several product cycles, which may translate to saving millions of dollars for the OEM. Therefore, the optimization of signal-to-frame packing in CAN-FD is the major focus of this dissertation. Another challenge addressed in this dissertation is the reliable mapping of a task model onto a given architecture, such that the end-to-end latency requirements are satisfied. This avoids costly redesign and redevelopment due to system design errors.

Design Space Exploration for Embedded Systems in Automotives

Prachi Joshi

(GENERAL AUDIENCE ABSTRACT)

Automobiles today are equipped with a variety of advanced features, such as adaptive cruise control, lane departure warning systems, information and entertainment systems, etc. These advanced features rely on electronics and software. A modern automobile consists of up to 100 computer systems that are interconnected by several buses (in-vehicle communication networks), exchanging thousands of signals (which are data entities such as sensor data, control commands, etc.). The addition of new functionalities means additional complexity and more demand of existing resources such as bus bandwidth. The automotive companies face a number of challenges such as reliability, safety and cost to incorporate the growing features in vehicles with the limited resources. In this dissertation, we study the problem of optimization of bandwidth utilization (BU) over a communication bus used in automotives. In Chapter 3, we show that for an automobile company even a modest improvement in BU of about 2.3% could potentially extend the bus architecture's lifetime by several product cycles. This may translate to saving millions of dollars for the company. Therefore, the optimization of bandwidth utilization over a communication bus is the major focus of this dissertation. Another problem addressed in this dissertation is the reliable mapping of a software model onto a given architecture (for an automotive system), such that the timing requirements are satisfied. This avoids costly redesign and redevelopment due to system design errors.

Dedicated to my parents

Acknowledgments

First and foremost, I wish to express my gratitude to my advisor Dr. Haibo Zeng whose skillful guidance, innovative ideas, great knowledge and experience have been instrumental in the course of my PhD. Although I had the opportunity of working with him for less than 3 years, I learnt a great deal from him and grew as a researcher under his mentorship. He not only shaped my end goals towards my PhD completion but also guided me in achieving them in a relatively short period of time. Without him this dissertation would not have been possible. His teachings shall remain ingrained lifelong. I wish to express my sincere gratitude to Dr. Sandeep Shukla, with whom I started my PhD journey. He motivated my interest in the research group 'CESCA' (Center for Embedded Systems for Critical Applications). I learnt a lot from him; his knowledge and intellect are awe-inspiring. I thank him for his invaluable guidance and necessary financial support. I wish to thank Dr. S.S. Ravi whose selfless support and encouragement has been of tremendous help during the course of my PhD. He is an inspiration to me and my friends. I also would like to thank the rest of my doctoral committee members Dr. Michael S. Hsiao, Dr. Yaling Yang and Dr. Anil Vullikanti for their time and their valuable feedback and suggestions on my dissertation. I would like thank my pillars of strength; my parents and sister for their endless love and support in all my endeavors. I extend thanks to: Dr. Unmesh D. Bordoloi and Dr. Soheil Samii, who were a great help on technical discussions about our work, everyone at General Motors for making my internship fruitful and enjoyable; Dr. Lakshmi Narasimhan for his support and camaraderie; Dr. Kalyani Ramchandran for her friendship. I thank my friends and lab mates: Udayasree Datla, Urvi Desai, Avik Dayal, Dr. Srivats Shukla, Dr. Mahesh Nanjundappa, Dr. Hamideh Bitaraf, Yecheng Zhao, Quingyu Liu, Vedahari Narasimhan G.,

Akshay Shastry, Shamit Bansal and Vibhav Nanda for their help and support. Thanks to Manisha and Akshay Sharma, Sanjay, Ishita, Vidushi and Sanjita Vyas, Sugandha Sharma, Abhiram Sharma who made my PhD journey bright and easier. I am grateful to my entire family for their support, especially my aunt, Karuna Sharma who helped me stay positive and confident.

Contents

List of Figures

List of Tables

List of Abbreviations

ABS Anti-lock Braking System

ACC Adaptive Cruise Control

ADAS Advanced Driver Assistance Systems

AFDX Avionics Full-Duplex Switched Ethernet

AUTOSAR AUTomotive Open System ARchitecture

BCTT Best Case Transmission Time

BL Bottom Left

BPP Bin Packing Problem

CAN Controller Area Network

CAN-FD Controller Area Network with Flexible Data-Rate

ECU Electronic Control Unit

ET Event-Triggered

FIFO First In First Out

I-PDU Interaction Layer Protocol Data Units

ILP Integer Linear Programming

JSSP Job Shop Scheduling Problem

LIN Local Interconnect Network

MILP Mixed Integer Linear Processing

MMP Makespan Minimization Problem

MOST Media Oriented Systems Transport

MSP Mapping-Scheduling Problem

OEM Original Equipment Manufacturer

SMT Satisfiability Modulo Theory

TDMA Time Division Multiple Access

TSN Time Sensitive Networking

TT Time-Triggered

TT-CAN Time Triggered-Controller Area Network

TTE Time Triggered Ethernet

TTP Time Triggered Protocol

WCET Worst Case Execution Time

WCTT Worst Case Transmission Time

Chapter 1

Introduction

Embedded systems are integral components of many safety critical applications such as automotive, avionics, medical instrumentation, etc. The challenges in the design and implementation of embedded systems have been increasing in recent years due to the growing demand for advanced and complex functionalities and shrinking time to market windows [61],[60]. Therefore, it is important to be able to develop the desired system in a time and cost efficient manner.

Embedded systems are ubiquitous in today's automotive systems because of advanced functionalities and replacement of some mechanical or hydraulic systems by electronic systems and software. High-end cars nowadays have more than 100 Electronic Control Units (ECUs) running various real-time and non real-time applications which need to coordinate with each other for various functions [48]. In such a complex setting, the end-to-end latency requirements imposed by the system specifications result in strict deadlines for completing many required control functions. Ensuring that the system meets the various deadline constraints makes the design and implementation tasks extremely challenging. Also the ECUs communicate with each other through thousands of signals sent over communication buses such as Controller Area Network (CAN), FlexRay, Time Triggered Ethernet (TTE), etc. which have distinct characteristics (time-triggered, event-triggered or heterogeneous). Several real time issues, such as message delay, jitter and task execution times need to be considered carefully while designing the system. In this dissertation, we tackle some of these challenges which

arise in the context of the **automotive embedded system design**.

Original Equipment Manufacturers (OEMs) (such as General Motors (GM), Toyota, etc.) and suppliers (such as Denso, Delphi, etc.) face a number of challenges in the system design phase as a result of the growing complexity and advancement in this domain. The discovery of design errors late in the development cycle is costly and it is even worse when such a discovery happens after the product is released in the market. For example, many Toyota vehicles were recalled in 2010 for unintended acceleration [36], which even caused loss of life. Thus finding bugs in the system design, architecture and usage of shared resources at the earliest stages of design is important.

Temporal guarantees are one of the key factors in ensuring the functional reliability and safety of the system. However, in addition to meeting all the functional requirements OEMs also have resource optimization considerations for the efficient development of reliable systems. Modern automotives are being developed with sophisticated and advanced features such as advanced driver assistance systems (ADAS), anti-lock braking system (ABS), power steering, etc. The addition of features requires more tasks and signals (that facilitate inter-task communication) which in turn requires additional ECUs and bandwidth resources. The increase in resources results in additional cost and space considerations. Thus the desired system development requirements also focus on optimizing the available resources in addition to meeting the functional requirements. In particular the focus of this dissertation is on **real-time communication protocols** and the real-time issues pertinent to them.

1.1 Overview of Communication Protocols for Automotive Embedded Systems

Communication networks form a key enabling technology for the integration and connection of the several electronics and software systems present in modern automotives. The real-time communication networks are mainly categorized based on their messaging[1] mechanism : 1) Event-triggered (ET), 2) Time-triggered (TT) and 3) TT and ET. In the ET category, the frames are sent on the successful completion of an event, (e.g., transmission of other frames), whereas in TT, the frames are transmitted at a fixed time (using a pre-defined schedule). CAN, CAN with Flexible Data-Rate (CAN-FD), Avionics Full-Duplex Switched Ethernet (AFDX) are ET based communication networks while Time Triggered CAN (TT-CAN) and Time Triggered Protocol (TTP) are TT communication networks. The third category networks have a provision for both TT and ET frame transmission. Time Triggered Ethernet (TTE) and FlexRay belong to this category [79].

The transmission mechanism of the communication network makes a huge impact on the temporal performance of the system. The performance metrics are: (i) *frame response time/latency* which is defined as the time when the frame is ready to the time it reaches its destination, and (ii) *jitter* which is the difference between the worst-case and best-case latencies [79]. Due to the different frame transmission mechanisms there are different scheduling and transmission problems that arise for each category. In TT communication, each frame is assigned bandwidth at a pre-defined time when the global clock reaches that time. Therefore, a deterministic schedule is set up on the network. The response time and jitter for such a communication can be easily computed from the pre-defined schedule, irrespective of the rest of the network. For ET communication, delays on the network may occur due to

[1]In this dissertation we use the term **frame** to denote a **message** . A frame is basically an entity which consists of a group of signals and it is transmitted on the network.

conflicts on the media or queuing mechanisms.

This dissertation mainly discusses the design exploration problems related to (CAN) /CAN-FD network optimization and mapping a given task model onto the TTE architecture. Therefore, a brief description of the two protocols is given in the following sections.

1.1.1 Controller Area Network (CAN)

Since its development in the mid 90s, the Controller Area Network (CAN) has attracted a significant amount of research from the real-time systems community. CAN is a broadcast digital bus which was designed to operate at a speed ranging from 20kbit/s to 1 Mbit/s [79]. The CAN protocol adopts a collision detection and resolution scheme, where the frame to be transmitted is chosen according to its identifier. When multiple nodes need to transmit over the same bus, the lowest identifier frame is selected for transmission. This arbitration protocol allows encoding the frame priority into the identifier field and implementing priority-based scheduling.

In recent years, automotive features keep growing in number and complexity, thereby demanding an increase in bandwidth requirements of the communication network resources. In order to bridge the gap between CAN and other higher data rate communication protocols (such as TTEthernet, MOST150, etc.) two major improvements were added to CAN to develop CAN-FD [25] : 1) the increase of bit-rate (of up to 8 Mbps) and 2) the increase of payload sizes (up to 64 bytes). The physical layer of CAN was unchanged: it still uses a bitwise arbitration method of contention resolution based on frame identifiers.

CAN-FD Frame Format and Bus Arbitration

The CAN-FD frame format is shown in Figure 1.1. Like CAN, a **dominant** bit is a logical 0 and a **recessive** bit is a logical 1. As in the figure, a CAN-FD frame is partitioned into two phases: arbitration phase and data phase.

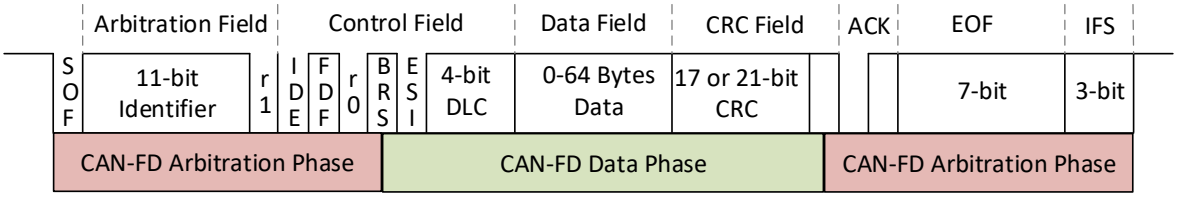


Figure 1.1: CAN-FD Frame Format (from [25]).

Arbitration Phase. The arbitration phase in the CAN-FD frame contains the following fields: SOF (Start Of Frame), arbitration, part of the control field, ACK (Acknowledgment), EOF (End OF Frame), and IFS (Inter-Frame Space). The 11-bit (or 29-bit in case of extended format) identifier represents the priority of the frame: the lower the value of the identifier, the higher the priority. The arbitration for transmission happens as follows. During the idle state of the bus, all the nodes with some ready frames send the 11-bit identifier after the SOF bit. During the transmission of the identifier bits, if a node transmits a recessive bit but finds a dominant bit on the bus, it stops transmission due to the presence of a higher priority node contesting for transmission. In the end, the node with the highest priority message wins the arbitration and continues the transmission.

The transmission of bits in the arbitration phase occurs at the arbitration bit-rate, and the duration of transmission for each bit is denoted as t_{arb}. For example, if the arbitration rate is chosen as 500 Kbps, then $t_{arb} = 2\mu s$.

Data Phase. The BRS (Bit-Rate Switch) bit is one of the additions to the CAN-FD frame format. It is used to decide whether the bit-rate in the data phase is the same as that of

the arbitration phase (BRS = 0) or it switches to the increased bit rate (BRS = 1). For CAN-FD the BRS bit in the frames is recessive (i.e., BRS = 1). At the increased rate of data transmission, each bit transmission occurs with a duration denoted by t_{data}. For example, if the data rate is chosen as 2 Mbps, $t_{data} = 0.5\mu s$. The 4-bit DLC (data-length code) field specifies the payload size (in bytes) of the data field. CAN-FD offers 16 distinct payload sizes: 0 through 8, 12, 16, 20, 24, 32, 48 and 64 bytes.

The data field is followed by the Cyclic Redundancy Check (CRC) field, which has 17 bits for payloads up to 16 bytes, and 21 bits otherwise. The CRC delimiter bit (recessive) is transmitted next. After this, the bit rate is changed back to that of the arbitration phase.

Bit Stuffing

CAN protocol follows the **bit stuffing** rule: If there are five dormant or recessive bits transmitted consecutively, then there is provision in the controller to transmit one *stuff bit* which is opposite in value to the five consecutive bits. This mechanism is provided in the receivers to detect errors on the bus. The bit stuffing mechanism is included in CAN-FD as well, however there are minor differences: The bit-stuffing rule is only applied from the start of a CAN-FD frame to the end of the payload section. For the CRC field, stuff bits are static and independent of the actual CRC value. For a 17-bit CRC, 4 stuff bits are added (making the CRC to be 21 bits in total), whereas for a 21-bit CRC, 5 stuff bits are added (making a total size of 26 bits) [5].

CAN-FD Timing Analysis

Using a bit stuffing scheme and the above described fields for CAN-FD, [5] provide expressions for the best-case (BCTT) and worst-case (WCTT) transmission times for CAN-FD frames. If p denotes the payload of a CAN-FD frame (in bytes), the expressions for BCTT

and WCTT are as follows.

$$\mathrm{BCTT}(p) = 29\,t_{arb} + \left(27 + 5\left\lceil\frac{p-16}{64}\right\rceil + 8p\right)t_{data} \tag{1.1}$$

$$\mathrm{WCTT}(p) = 32\,t_{arb} + \left(28 + 5\left\lceil\frac{p-16}{64}\right\rceil + 10p\right)t_{data} \tag{1.2}$$

To predictably guarantee that the frames transmitting on CAN bus meet their deadlines, the early analysis on the CAN frame response time was derived by Tindell et al. using an analogy to the CPU scheduling results [69]. The analysis was later found to be flawed by Davis et al. and they provided a set of formulas for the exact evaluation and safe approximation of the worst case frame response times [10].

Timing Analysis

The worst-case response time for a frame F_i is always inside the busy period. The busy period of priority level-i is a contiguous interval of time that starts at the critical instant, during which any frame of priority lower than F_i is unable to win arbitration. The length of the busy period $L(F_i)$ and the index $q^{\max}(F_i)$ of the last instance are calculated as

$$L(F_i) = B(F_i) + \sum_{j\in hp(i)\bigcup\{i\}}\left\lceil\frac{L(F_i)}{T(F_j)}\right\rceil C(F_j), \qquad q^{\max}(F_i) = \left\lceil\frac{L(F_i)}{T(F_i)}\right\rceil \tag{1.3}$$

where $hp(i)$ is the set of frames with priority higher than F_i, and $B(F_i)$ is the *blocking time*, i.e., the maximum time spent on waiting for the transmission of a lower priority message already on the bus when F_i becomes ready. $C(F_j)$ is the worst case transmission time (WCTT) for frame F_j and $T(F_j)$ is the period of F_j.

The response time $R(F_{i,q})$ of the q-th instance $F_{i,q}$ in the busy period is given by

$$R(F_{i,q}) = w(F_{i,q}) - (q-1)T(F_i) + C(F_i) \tag{1.4}$$

where q ranges from 1 to the last instance $q^{\max}(F_i)$ of F_i inside the busy period. The worst-case queuing delay $w(F_{i,q})$ for the q-th instance in the busy period is

$$w(F_{i,q}) = B(F_i) + (q-1)C(F_i) + \sum_{j \in hp(i)} \left\lceil \frac{w(F_{i,q})}{T(F_j)} \right\rceil C(F_j) \tag{1.5}$$

In Equation (1.5), $w(F_{i,q})$ appears on both sides. However, the right hand side is a monotonic non-decreasing function of $w(F_{i,q})$. Hence, $w(F_{i,q})$ can be solved using the iterative procedure defined by the equation below.

$$w^{n+1}(F_{i,q}) = B(F_i) + (q-1)C(F_i) + \sum_{j \in hp(i)} \left\lceil \frac{w^n(F_{i,q})}{T(F_j)} \right\rceil C(F_j) \tag{1.6}$$

The calculation can start with an initial value of $w^0(F_{i,q}) = B(F_i) + (q-1)C_i$, and stop when $w^{n+1}(F_{i,q}) = w^n(F_{i,q})$ or $w^{n+1}(F_{i,q}) - (q-1)T(F_i) + C(F_i) > D(F_i)$, the latter condition indicating that F_i is unschedulable. The worst-case response time of F_i, denoted by $R(F_i)$, is the maximum among all its instances in the busy period; that is,

$$R(F_i) = \max_{q=1,\ldots,q^{\max}(F_i)} \{R(F_{i,q})\} \tag{1.7}$$

1.1.2 Time Triggered Ethernet (TTE)

Packet-switched Ethernet are gaining popularity in the next-generation automotive communication architectures. Due to the superior bandwidth of Ethernet, it is ideal for the high

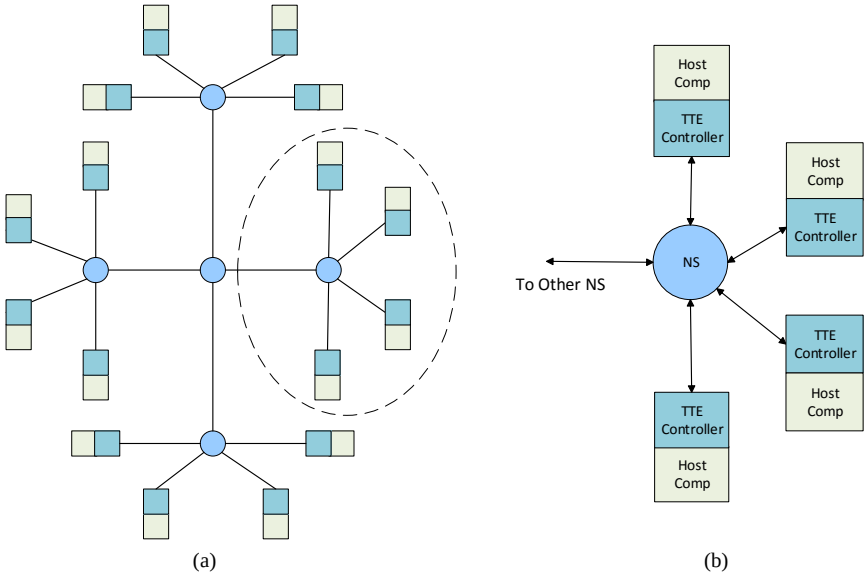


Figure 1.2: Time-Triggered Ethernet System: (a) Snowflake Topology; (b) The architecture cluster in the dashed circle of (a).

communication demands of systems such as ADAS, infotainment, etc. As a switched network, Ethernet provides a scalable, high-speed, and cost-effective communication platform, which allows arbitrary topologies [79].

Deterministic Ethernet is a class of network communication technologies which is used to integrate real-time traffic along with the regular Ethernet traffic on IEEE 802 Ethernet [70]. Time Sensitive Networking (TSN) and Time Triggered Ethernet (TTE) by TTTech [70] are two important incarnations which support Deterministic Ethernet. In this dissertation we consider a TTE (developed at the Vienna University of Technology [38]) based architecture, which provides a deterministic and collision free communication for time-triggered frames using a static time division multiple access (TDMA) scheme. It has a higher bandwidth (1Gbps) than the other time-triggered communication solutions like FlexRay.

A TTE network contains a set of clusters and each cluster consists of a set of network switches (NS) and a set of computation units called end systems (ES). NS is responsible

for the integration of different traffic on the TTE as well as for the clock synchronization of the cluster. Each cluster has global synchronization and the TTE schedules are derived per cluster. The set of clusters can take up star, tree or snowflake topologies. An example of a TTE network having snowflake topology is shown in Figure 1.2(a), where the rectangles denote the end systems and the circles represent network switches. Each cluster is connected to other clusters via a central switch. Figure 1.2(b) represents a TTE architecture cluster with four ESes and one NS.

TTE transmits signals over the network after packing them into frames. The end systems and network switches are connected via links which are full duplex. TTE transmits three kinds of traffic over the network: *time triggered* (TT), *rate constrained* (RC), and *best effort* (BE). TT communication is deterministic and hence it is used for critical applications such as brake-by-wire. RC communication is used for applications which have relatively relaxed constraints for determinism (e.g., audio or video streaming). BE frames have less priority than both the TT and RC frames and use the remaining bandwidth. These are non-deterministic as whether and when these frames would be received is not known. Frames pertaining to the maintenance and configuration phases can be taken as examples of BE frames. Therefore, TTE is highly suited for applications with different levels of criticality.

For Ethernet the contention among frames occurs at the switches. Therefore, for worst case response time computation, the topology of the system and the resource allocation should be identified. There are multiple delays that can occur inside a switch, such as *input queuing delay*, *forwarding delay* and *output queuing delay*. The input and forwarding delay are often approximated by constant terms. The output queuing delay depends on the interference from other frames at the switch port, which in turn depends on the scheduling policy of the switch. The transmission latency of a frame includes all timing delays introduced due to the interfering traffic streams. There are different classes of traffic in the switches which

signify the priority level. The frames are ordered into a set of First In First Out (FIFO) queues corresponding to each priority level. Thus, the queuing delay also depends on the arrival time of a frame (for frames of the same priority). A Static-Priority Non-Preemptive (SPNP) scheduler manages the FIFO queues. In order to compute the worst-case queuing delay for a frame, all blocking effects, which can occur in this combination of FIFO and SPNP scheduling, must be considered [79].

1.2 Design Space Exploration Challenges in Automotives

1.2.1 CAN-FD Bandwidth Optimization

Given the payload size p of a frame F and period $T(F)$, the bandwidth utilization for the frame on the network is given by

$$BU(F) = \frac{WCTT(p)}{T(F)} \tag{1.8}$$

It is evident from Equation (1.8) that **Bandwidth Utilization**(BU) of a frame essentially quantifies the usage of the network by a frame F in one cycle. BU is often reported in percentage for a fair comparison. The total BU (on a bus) is given by the summation of the individual BU of each frame. Although in general a higher BU on a bus signifies an efficient usage of the network, it also implies **saturation of the bus** and difficulty in transmitting more signals. For a shared bus it also indicates the inability in meeting the deadline constraints of each signal.

In order for all the available frames to meet their deadlines, it is required that their worst

case response times are smaller than their respective deadlines. From the CAN/CAN-FD response time analysis it is clear that the response time of a frame depends on 1) its worst case transmission time and 2) the worst case transmission time of the higher priority frames.

From Equations (1.2) and (1.8) and the response time analysis we observe that both the bandwidth utilization and the response time of a frame depend on the worst-case transmission time, which in turn depends on the signals packed into the frame. For a given set of signals there are many possible ways of packing them into CAN-FD (and CAN) frames, resulting in different utilizations and schedulability. The assignment of a set of signals with different sizes and periods to a finite number of frames is known as the ***frame packing problem***. In the literature this problem is often compared to the bin packing problem which is known to be NP-hard [19]. Therefore it is evident that investigating the frame packing problem is important to optimize the bus bandwidth and ensuring the temporal requirements of the signals are satisfied.

Optimizing Frame Packing in CAN/CAN-FD

Even today the signal-to-frame packing in CAN/CAN-FD is done manually. This gives a packing which satisfies legacy constraints and is easier for engineers to trace and change when required. However such a manual packing is often not optimal and may waste bandwidth on the bus. Such a waste of bandwidth is undesirable since with the growing number of signals there is more demand on the bus bandwidth. Also modern features such as ACC have stringent timing requirements. Due to higher bus loads, erroneous transmissions have been observed to increase. For example in [12] it is shown that when bus loads exceed 50%, the reliability of signal transmission is not guaranteed. This may lead to system failures and hence needs to be avoided at all cost. The need for optimizing CAN/CAN-FD networks has become an essential industrial requirement due to the following reasons [14]:

1. It helps in optimizing the hardware costs, weight, space, etc.
2. It provides better system extensibility (ability to add new features).
3. It leads to better communication performance and helps in meeting the required timing guarantees for features with stringent deadline constraints.

In order to optimize the CAN/CAN-FD network the bus bandwidth wastage needs to be minimized. The wastage (on a single bus) may be due to (1) unused space in the CAN-FD frame or (2) signals of different periods packed in the same frame. Although it is ideal to have zero unused space in a frame packing, since CAN-FD frames have discontinuous sizes (as discussed in Section 1.1.1), it is common to have some unused space in the frames. One of the design criteria to optimize bus bandwidth could be to minimize the unused space. The second criterion that would reduce loss of bandwidth is packing signals with similar timing requirements in a frame. Also if there are multiple CAN-FD buses and there is inter-bus transmission of signals, then the designer needs to consider the destination bus of signals in order to determine an optimal frame packing. These problems and our proposed solutions are discussed in detail in Chapters 2 and 3.

1.2.2 Mapping Task Model Onto Architecture Model

The set of functionalities provided by a system are given as specifications which define the electronic/software system of an automotive. The physical architecture captures the topology of the automotive including the Electronic Control Units (ECUs) and the communication buses such as Controller Area Network(CAN), FlexRay, etc. [81]. The deployment of functional model onto the physical architecture model is a design decision which must be taken by the designer in such way that the mapping satisfies timing requirements such as end-to-end latencies, task and signal deadlines, etc. The mapping or task allocation problem can

often be translated into the *module allocation problem* (depending on the constraints of the problem). This is a fundamental combinatorial problem which is shown to be NP-hard [13]. The task and signal allocation must be done with a focus on the reliability of meeting functional as well as timing guarantees. The methods/tools for allocation may include a fault detection scheme (e.g. fault tree generation) and/or a fall-back method for correction.

In this dissertation we consider the problem of mapping behavioral (task) model onto an architectural model in order to optimize end-to-end latency in executing a distributed function on the specified platform architecture. The objective is to ensure the schedulability of tasks and signals on the given architecture, and satisfy the desired end-to-end path latency constraints.

1.3 Outline and Summary of Contributions

In this thesis we have studied frame packing problem for CAN-FD and task mapping and frame scheduling problem for TTE. Chapter 2 discusses the signal to frame packing problem for a multi-domain CAN-FD system, Chapter 3 presents the signal offset assignment problem to improve the frame packing in CAN-FD. Chapter 4 considers the problem of mapping of a functional model onto an architectural model for an automotive. The architecture is based on a TTE network here. Finally Chapter 5 concludes the dissertation and presents directions for future work in this area.

1.3.1 The Multi-Domain Frame Packing Problem for CAN-FD

In Chapter 2 we focus on CAN-FD technology and address the problem of frame packing for a system with multiple domains connected via CAN-FD buses. The frame packing

problem in CAN and CAN-FD has been addressed in the literature; however, unlike our work, only a single sub-system has been considered. The proliferation of cross-domain traffic can significantly contribute to bandwidth bottlenecks, and as we show in this work, it leads to a non-trivial optimization problem called as the *multi-domain frame packing problem for CAN-FD*. Even for a single sub-system (domain), the problem is already challenging. In the literature its relationship to the bin packing problem is shown which is known to be NP-hard. The multi-domain aspect adds to the complexity of the problem.

In this work, we consider multiple CAN-FD domains (buses) that are connected by a central gateway. We study two models of the central gateway: 1) Basic Central Gateway Model (BCGM) and 2) Advanced Central Gateway Model (ACGM) and develop a two-stage optimization procedure for each model. For BCGM, we propose an ILP based approach to generate an optimal solution for frame packing as well as a heuristic that scales to large problem sizes for the first stage. In the second stage, we propose an extension to Audsley's algorithm [2] for optimal priority assignment with multi-domain frames. In case the priority assignment does not lead to a feasible solution, we provide an effective strategy to re-pack the frames. We have presented the multi-domain frame packing problem in CAN-FD with a BCGM in [32].

For ACGM we propose four different heuristics for the first stage of signal-to-frame packing and frame decomposition at the gateway. In the second stage we do a schedulability analysis for the packed frames and again use Audsley's algorithm to ensure that they all meet the deadline requirements.

We conduct experiments on synthetic systems (whose characteristics follow the guidelines of real-world automotive benchmarks) and show the efficiency and performance of our heuristics. For the BCGM we show that our heuristic runs extremely fast compared to the computationally expensive (in terms of both time and memory) ILP and yet returns solutions that

are on average within 3% of those produced by the ILP in terms of bandwidth utilization per domain. Our experiments also show that the repacking strategy is effective in that it often leads to schedulable solutions. Compared to the approach that ignores cross-domain considerations, our approach can typically save 6%-10% bandwidth utilization per domain.

Since the ACGM is an improved and advanced gateway model, we further conduct experiments to show the advantage of using the ACGM and compare the results with the BCGM approaches. We observe about 4.5% improvement in bandwidth utilization per domain by using ACGM over BCGM. We also observe 11% percent improvement in schedulability.

1.3.2 Offset Assignment to Signals for Improving Frame Packing in CAN-FD

In continuation with the goal of minimizing the bandwidth utilization of the CAN-FD bus, we explore more sophisticated techniques to further optimize the bandwidth in order to do a signal-to-frame packing. In Chapter 3 we look into the concept of using an *offset assignment* to signals in order to save bandwidth loss occurring due to the different frequencies of the signals.

We propose and formulate, the *signal offset assignment problem* (SOAP) in a frame in order to improve the bus bandwidth utilization. We prove that SOAP is NP-complete and propose a general approximation framework (GAF) for SOAP which can use any approximation algorithm for the makespan minimization problem (MMP) in multiprocessor systems. We derive the performance guarantee provided by GAF as a function of the performance guarantee of the approximation algorithm for MMP and the number of signal periods in the frame. We demonstrate the efficacy of our approach through experiments using three different algorithms (two approximation algorithms and an integer linear programming formulation) for

MMP in GAF.

In addition to this we propose two more heuristics that further improve the bandwidth utilization of a single bus by using offset assignment. Our results indicate that by using offsets for signals in frame packing schemes, one can achieve about 11.44% improvement in bandwidth utilization (on a single bus) in CAN-FD systems. We also propose a novel transmission scheme to implement SOAP on AUTomotive Open System ARchitecture (AUTOSAR)[2]. Further we extend our work on offset assignment to signals and propose a 2-dimensional strip packing based approach for offset assignment. We apply this approach to frame packing and obtain a modest improvement in bandwidth utilization. However the potential of this proposed approach is not fully utilized (requires more experiments) and hence, it is the main focus of our future work.

1.3.3 Design Space Exploration for Time Triggered Ethernet-based Architecture of Automotive Systems

In Chapter 4 we address the problem of task mapping and communication scheduling in automotive design. The system model is specified using a dataflow task communication model and the target architecture is based on TTE. The design variables are the mapping of tasks onto the end systems (processors) in the architecture and the scheduling of all time-triggered frames such that they meet their respective deadlines. The constraints include the schedulability of tasks and signals, as well as the latency constraints of the critical paths as specified by the designer. We show that the problem is NP-hard by a reduction from the job shop scheduling problem. We also develop a heuristic to solve this problem. The heuristic contains four steps and all of them (except scheduling) are formulated using Integer Linear

[2]AUTOSAR is a de-facto open industry standard for an automotive software architecture

Programming (ILP). We present experimental results on an industrial benchmark and two synthetic benchmarks which show the efficiency and scalability of our approach.

The following is the list of publications during the course of my PhD:

Conferences:

1. **Joshi, P.**, Ravi, S. S., Samii, S., Bordoloi, U. D., Shukla, S., & Zeng, H. (2017, December). Offset Assignment to Signals for Improving Frame Packing in CAN-FD. In Real-Time Systems Symposium (RTSS), 2017 IEEE (pp. 167-177). IEEE.

2. **Joshi, P.**, Zeng, H., Bordoloi, U. D., Samii, S., Ravi, S. S., & Shukla, S. K. (2017, June). The Multi-Domain Frame Packing Problem for CAN-FD. In Euromicro Conference on Real-Time Systems (ECRTS), 2017.

3. **Joshi, P.**, Zeng, H., Shukla, S. K., Lin, C. W., & Yu, H. (2016, October). Design space exploration for deterministic ethernet-based architecture of automotive systems. In High Level Design Validation and Test Workshop (HLDVT), 2016 IEEE International (pp. 53-61). IEEE. (Invited)

4. **Joshi, P.**, Shukla, S. K., Talpin, J. P., & Yu, H. (2015, April). Mapping functional behavior onto architectural model in a model driven embedded system design. In Proceedings of the 30th Annual ACM Symposium on Applied Computing (pp. 1624-1630). ACM.

5. Yu, H., **Joshi, P.**, Talpin, J. P., Shukla, S., & Shiraishi, S. (2015, June). The challenge of interoperability: model-based integration for automotive control software. In Proceedings of the 52nd Annual Design Automation Conference (p. 58). ACM.

6. Yu, H., Talpin, J. P., Shukla, S., **Joshi, P.**, & Shiraishi, S. (2014). Towards an architecture-centric approach dedicated to model-based virtual integration for embed-

ded software systems. In ACVI 2014-Architecture Centric Virtual Integration Workshop Proceedings (p. 79).

Journals and Book Chapter:

1. **Joshi, P.**, Zeng, H., Bordoloi, U. D., Samii, S., Ravi, S. S., & Shukla, S. K. "Offset Assignment Approaches to Signals for Improving Frame Packing in CAN-FD" In preparation.

2. **Joshi, P.**, Zeng, H., Bordoloi, U. D., Samii, S., Ravi, S. S., & Shukla, S. K. "The Multi-Domain Frame Packing Problem in CAN-FD: Formulation, Efficient Heuristics and Benefits of Using an Advanced Gateway". Real Time Systems Journal (RTSJ). Submitted.

3. Zeng, H., **Joshi, P.**, Thiele, D., Diemer, J., Axer, P., Ernst, R., & Eles, P. (2017). Networked Real-Time Embedded Systems. Handbook of Hardware/Software Codesign, 753-792.

Patents:

1. **Joshi, Prachi**, et al."Method and System for Transmission of Signals with Efficient Bandwidth Utilization." Filed February, 2018.

2. **Joshi, Prachi**, et al. "Bottom-up approach for integrating models for software components using contracts." U.S. Patent No. 9,477,446. 25 Oct. 2016.

Chapter 2

The Frame Packing problem for Multi-Domain CAN-FD Network

2.1 Introduction

The Controller Area Network with Flexible Data-Rate (CAN-FD) is a new communication protocol to meet the bandwidth requirements for the ever growing volume of data exchanged over the embedded network inside modern vehicles. The problem of frame packing for CAN-FD, as studied in the literature, assumes a single sub-system where one CAN-FD bus serves as the communication medium among several Electronic Control Units (ECUs). Modern automotive electronic systems, on the other hand, consist of several sub-systems, each facilitating a certain functional domain such as powertrain, chassis and suspension. A substantial fraction of all signals is exchanged across sub-systems. In this work, we study the frame packing problem for CAN-FD with multiple sub-systems connected by a gateway. We consider two types of gateways: 1) A "basic" gateway model which simply forwards the incoming frames to the respective destination domains, and 2) An "advanced" gateway model which can decompose the incoming frames: by first grouping the individual signals (based on their destination domains) and then packing each group of signals into a new frame. This is done in order to further reduce the bandwidth utilization on the network. We propose a two-stage optimization framework for frame packing for the system. In the first stage,

we pack the signals into frames with the objective of minimizing the bandwidth utilization. In the second stage, we extend Audsley's algorithm to assign priorities/identifiers to the frames. In case the resulting solution is not schedulable, our framework provides a potential repacking method. For the basic gateway model we propose two solution approaches for the first stage: (a) an Integer Linear Programming (ILP) formulation that provides an optimal solution but is computationally expensive for industrial-size problems; and (b) a greedy heuristic that scales well and provides solutions that are comparable to optimal solutions. For the advanced gateway model we propose and develop 4 heuristics for addressing the signal-to-frame packing in a CAN-FD multi-domain system with a smart gateway.

We present our experimental results for both models of the central gateway and show the efficiency of our frame-packing frameworks in achieving feasible solutions with low bandwidth utilization. The results also show a significant improvement over the state-of-the-art. In the first stage, we pack the signals into frames with the objective of minimizing the bandwidth utilization. In the second stage, we extend Audsley's algorithm to assign priorities/identifiers to the frames. In case the resulting solution is not schedulable, our framework provides a potential repacking method. We propose two solution approaches: (a) an Integer Linear Programming (ILP) formulation that provides an optimal solution but is computationally expensive for industrial-size problems; and (b) a greedy heuristic that scales well and provides solutions that are comparable to optimal solutions. Experimental results show the efficiency of our optimization framework in achieving feasible solutions with low bandwidth utilization. The results also show a significant improvement over the case when there is no cross-domain consideration (as in prior work).

2.1.1 Related Work

As mentioned earlier, the frame packing problem has been considered in the literature only for a single domain in CAN-FD. In [5], a dynamic programming approach is presented for packing the signals followed by a priority assignment step. In [71], it is observed that schedulability of frames can be improved by packing same period signals in each frame. In [47], a single-step Integer Linear Programming (ILP) formulation is presented to achieve both optimal bandwidth utilization and schedulability. However, its applicability is limited to medium-size problems. In [81], the authors map signals to frames on CAN as part of their task allocation and priority assignment problem to optimize end-to-end latency using an MILP.

The frame packing problem is related to the classical bin packing problem (BPP) which is known to be NP-hard. For CAN-FD, a particularly relevant subclass of BPP is the *variable-sized bin packing problem* (VSBPP). While the classical bin packing problem has been studied extensively (e.g., [19]), VSBPP has received relatively less attention. [16] proposes and formally analyzes the performance of three heuristics for VSBPP. [46] presents a polynomial-time approximation scheme for VSBPP. We note that the approximation algorithms for VSBPP cannot be directly applied to the frame packing problem for CAN-FD since the goal of the latter problem is to minimize bandwidth utilization instead of the number of bins (frames). In addition, the frame packing problem must consider both the size and the period of each signal.

For standard CAN, both [53] and [59] present frame packing approaches inspired by the *next fit decreasing* heuristic for BPP. The difference is that the algorithm in [53] sorts the signals according to their periods, while the one in [59] sorts them based on their deadlines. [58] presents a frame packing heuristic which sorts the signals by their bandwidth utilization and

then packs this list of sorted signals alternately from both sides of the list (to increase the chances of signals with similar periods to be packed together).

The frame packing problem has also been considered under other communication protocols that are time-triggered (such as FlexRay static segment studied in [9, 33, 44, 67, 77]) or mixed event/time-triggered discussed in [55]. The nature of these communication protocols, and thus the frame packing problem, is very different from that for CAN and CAN-FD. Hence, the corresponding approaches and results are not directly applicable here.

For the multi-domain CAN-FD system connected by a central gateway, we propose our solutions by considering two models for the central gateway, a "basic" central gateway model whose functionality is to simply forward the incoming frames, and an "advanced" central gateway model which is more sophisticated and enables decomposition of the existing frames to the respective destination domains. There have been some work by researchers on modeling gateways for connecting CAN to higher bandwidth networks such as Ethernet, etc. In [26] the authors present a CAN to AVB Ethernet gateway, where multiple CAN frames are aggregated into a single AVB Ethernet frame. Similarly, [34] and [68] present a transformation from Ethernet networks to CAN based networks and vice-versa using a gateway and a formal timing analysis of the CAN-to-Ethernet gateway strategies, respectively. Since all these approaches deal with integrating CAN network to other networks we cannot directly use them for our problem of frame packing in the multi-domain CAN-FD system.

2.2 Motivational Examples

We saw that the overhead of a frame is one of the factors that must be considered while solving the frame packing problem. From Equation (2.6), it is clear that the period of the frame is also a factor that determines the bandwidth utilization. Since the period of a frame

Table 2.1: Signal parameters used in motivational examples

Signal	Size(Bytes)	Period(ms)	Destination Domain
σ_1	13	10	D_2
σ_2	7	10	D_3
σ_3	11	10	D_2
σ_4	5	10	D_1

F is taken to be the minimum of the periods of all the signals packed in F, bandwidth utilization is significantly impacted by how signals are packed into frames. For a multi-domain CAN-FD system, where signals from one ECU may be transmitted to more than one domain, it is also essential to consider the destination domain of each signal in optimizing bandwidth utilization. We now illustrate these points with the following example.

Example: Consider a CAN-FD system with three domains/subsystems denoted by D_1, D_2 and D_3. Let an ECU E_1 on D_1 produce signals $\sigma_1, \sigma_2, \sigma_3$ and σ_4 with sizes 13, 7, 5 and 11 bytes respectively. Let us assume that all the signals have the same period, namely 10 ms. Table 4.3 gives the signal parameters for this example.

Case 1: Pack all the signals into the same frame. In this case, we need a frame of size 48 bytes (since total signal size = 36 bytes). Therefore, the worst-case transmission time in this case is WCTT(48) = 320.5μs over each domain. The total bandwidth utilization of this frame over the entire system is the sum of the utilization over domains D_1, D_2 and D_3 which is equal to

$$\frac{\text{WCTT}(48)}{10,000} + \frac{\text{WCTT}(48)}{10,000} + \frac{\text{WCTT}(48)}{10,000} = 9.615\% \tag{2.1}$$

In this case, we observe that 12 bytes of the frame are wasted (owing to the discontinuous frame sizes available in CAN-FD). So, we consider another packing scenario which tries to better utilize the payload space.

Case 2: Pack the signals σ_1 and σ_2 in a frame F_1 of size 20 and σ_3 and σ_4 in a frame F_2 of

size 16. In this case, both the frames are completely utilized in terms of their payload size and no space is wasted. The worst-case transmission time of $F_1 = \text{WCTT}(20) = 180.5\mu s$ and that of $F_2 = \text{WCTT}(16) = 158\mu s$. D_1 is the source domain and hence both the frames must be transmitted over D_1. However, F_1 must also be transmitted to domains D_2 and D_3 (due to signals σ_1 and σ_2). Hence, the total utilization of frame F_1 over the system is given by

$$\frac{\text{WCTT}(20)}{10,000} + \frac{\text{WCTT}(20)}{10,000} + \frac{\text{WCTT}(20)}{10,000} = 5.415\% \tag{2.2}$$

Similarly the total utilization of frame F_2 is given by

$$\frac{\text{WCTT}(16)}{10,000} + \frac{\text{WCTT}(16)}{10,000} = 3.16\% \tag{2.3}$$

Thus, the total utilization due to both the frames is 8.7555%.

Case 3: We now consider a third option for packing the signals, where we still utilize the payload sizes of the frames to the maximum possible extent. However, this time we pack signals σ_1 and σ_3 in a frame F_1 of size 24 and signals σ_2 and σ_4 in a frame F_2 of size 12. Now, the total bandwidth utilization for F_1, computed over domains D_1 and $D2$ (since F_1 now has only signals for D_2), is given by

$$\frac{\text{WCTT}(24)}{10,000} + \frac{\text{WCTT}(24)}{10,000} = 4.01\% \tag{2.4}$$

Likewise, for F_2, the total bandwidth computed over D_1 and D_3 is given by

$$\frac{\text{WCTT}(12)}{10,000} + \frac{\text{WCTT}(12)}{10,000} = 2.76\% \tag{2.5}$$

In this case, the total utilization of the entire system is 6.77%.

Cases 2 and 3 above point out known techniques for the single domain frame packing problem

cannot be used directly to handle the multi-domain case. Further, in the above example, all the signals have the same period. In case of signals with different periods, the problem becomes even more complex since the period of a frame (which is the smallest of the periods of the signals in the frame) directly impacts bandwidth utilization.

Also, once the signals are packed into frames, we also need to ensure that the frames are schedulable on the network. The schedulability analysis is done by first assigning unique priorities to all the frames using an extension of Audsley's algorithm [2] for priority assignment. Each frame is scheduled on the system based on its priority. The response time analysis is based on a state-of-art technique for CAN networks [10]. We discuss these aspects in more detail in Section 2.7.1.

2.3 Problem Formulation

We assume a network topology where several CAN-FD sub-systems, typically serving different domains, are connected to a central gateway. We use the terms "domain" and "sub-system" interchangeably. The ECUs in each domain generate signals which must be packed into frames and transmitted to their destination domains. Each domain uses a CAN-FD bus for data communication. The gateway is responsible for forwarding the frames to their respective destination domains. Figure 2.1 provides an example of a system where three domains are connected by a gateway. We model the central gateway in two ways:

1. 'Basic Central Gateway Model' (BCGM): In this model the central gateway forwards the incoming frames to their destination domains without decomposing the frames or reassigning their frame identifiers. The gateway has no intelligence to unpack and separate frame signals (based on the destination domains) once they are packed. Such an architecture is common in the automotive industry [28]. While the simplicity of such

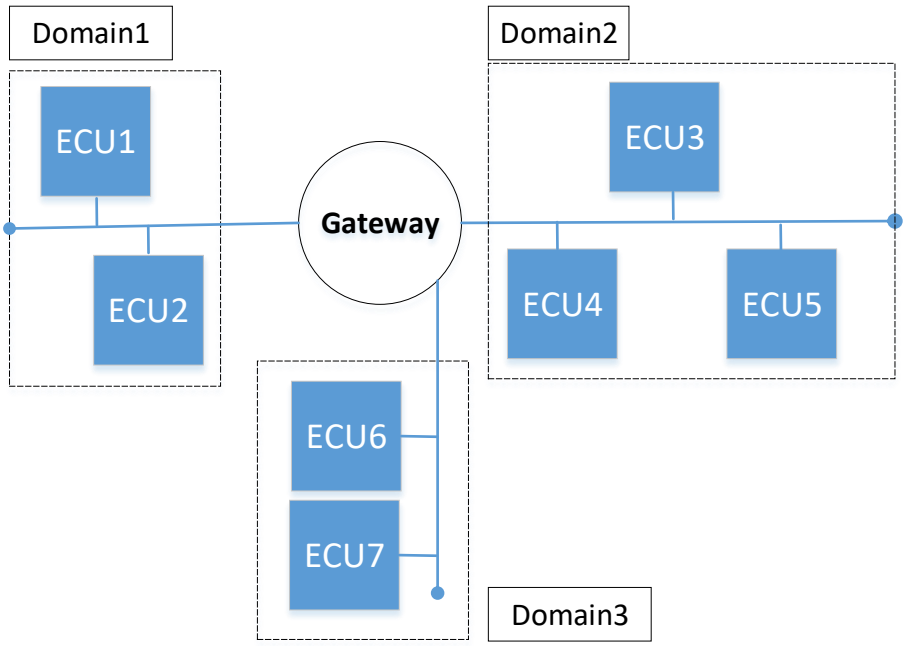


Figure 2.1: A multi-domain CAN-FD architecture

an architecture is an attractive property for automotive applications it does prohibit the optimal use of communication bandwidth due to the constraint that frames cannot be modified and the entire payload must be gated to their destination networks.

2. 'Advanced Central Gateway Model' (ACGM) : In this model the central gateway unpacks the incoming frames and reorganizes the constituent signals into new frames (with different frame identifiers than the original frame) based on their destination domains. They are then transmitted onto their corresponding destination buses. Figure 2.2 presents the proposed model of the advanced gateway. In this figure, when a frame F_1 (from domain D_1) enters the ACGM, it is separated into two new frames: F_2, consisting of σ_2, σ_3 and transmitted to D_2; and F_3, consisting of σ_4, σ_5 and transmitted to $D3$. In addition to that, the signal σ_1 (from source domain, D_1) is not transmitted further. We discuss the additional challenges introduced in the problem and our pro-

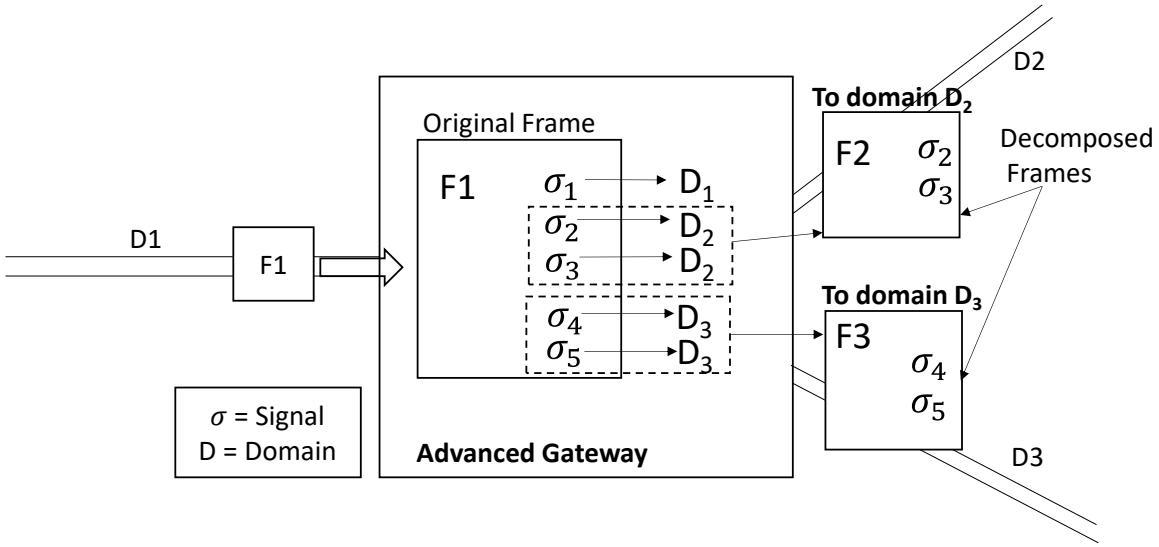


Figure 2.2: Proposed Advanced Central Gateway Model

posed heuristics in order to leverage the decomposition done by the advanced gateway model to further reduce bandwidth utilization in Section 2.7.

Problem Description: Let $\mathbb{D} = \{\Delta_1, \Delta_2, \ldots, \Delta_{|\mathbb{D}|}\}$ denote the set of *domains*. Let n_i denote the number of ECUs in domain Δ_i, $1 \leq i \leq |\mathbb{D}|$. The jth ECU from domain Δ_i is denoted by $\psi_{i,j}$. The set of signals generated by ECU $\psi_{i,j}$ is represented by $\mathcal{S}(\psi_{i,j}) = \{\sigma_{i,j}^k \mid k = 1, \ldots, |\mathcal{S}(\psi_{i,j})|\}$. Each signal $\sigma \in \mathcal{S}(\psi_{i,j})$ is specified as a quadruple $\langle\, t(\sigma), d(\sigma), p(\sigma), \delta(\sigma)\,\rangle$, whose components denote respectively the period, deadline, size (in bytes) and domains (including the source and destinations) of signal σ.

The output of the multi-domain frame packing problem is a set of frames $\Gamma = \{\gamma_1, \gamma_2, \ldots\}$ satisfying *all* of the following conditions.

1. Each signal σ is placed in exactly one frame γ.

2. For each frame γ, all the signals in γ are from the *same* ECU, and the periods of all

the signals in γ are **harmonic**[1] (i.e., $\forall \sigma_i, \sigma_j \in \gamma$, there exists an integer k such that either $t(\sigma_i) = k \cdot t(\sigma_j)$ or $t(\sigma_j) = k \cdot t(\sigma_i)$).

3. The sum of the sizes of all signals in a frame γ is at most the payload size of γ.

4. The packed frames are schedulable in all the domains in which they are transmitted.

Each frame γ is characterized by a tuple $\langle \mathcal{S}(\gamma), \Delta(\gamma), T(\gamma), D(\gamma), P(\gamma), C(\gamma), \pi(\gamma) \rangle$, where $\mathcal{S}(\gamma)$ is the set of signals packed into γ and $\Delta(\gamma)$ is the set of domains of the signals in γ. The quantities $T(\gamma)$, $D(\gamma)$, $P(\gamma)$, and $C(\gamma)$ are respectively the period, deadline, payload size (in bytes), and WCTT of the frame γ. Given $\mathcal{S}(\gamma)$, the other parameters of the frame γ are determined as follows.

- $\Delta(\gamma) = \bigcup_{\sigma \in \mathcal{S}(\gamma)} \delta(\sigma)$; i.e., the set of domains of γ is the union of those for all the signals in γ.

- $T(\gamma) = \text{gcd}\{t(\sigma) \ : \ \sigma \in \mathcal{S}(\gamma)\}$; i.e., the period of γ is the greatest common divisor (gcd) of the periods of the signals in γ.

- $D(\gamma) = \min\{d(\sigma) \ : \ \sigma \in \mathcal{S}(\gamma)\}$; i.e., the deadline of γ is the smallest deadline among the signals in γ.

- $P(\gamma) \geq \sum_{\sigma \in \mathcal{S}(\gamma)} p(\sigma)$; i.e., the payload of γ is large enough to contain its constituent signals. The CAN-FD standard [25] restricts $P(\gamma)$ to be one of the following values: 0 through 8, 12, 16, 20, 24, 32, 48 and 64 bytes.

- The WCTT $C(\gamma)$ of a frame γ is determined by Equation (1.2), with the variable p being replaced by $P(\gamma)$.

[1] Note that our approach is valid even without this condition.

- $\pi(\gamma)$ represents the unique priority (across all domains) assigned to frame γ.

The bandwidth utilization $U(\gamma)$ of frame γ is defined as

$$U(\gamma) = \frac{C(\gamma)}{T(\gamma)} \tag{2.6}$$

The objective is to minimize the total bandwidth utilization over all the domains. This is motivated by extensibility to accommodate possible future functions [5].

Table 2.2 summarizes the parameters of each signal and frame. In the following, we define the problem and review the schedulability analysis for CAN-FD.

Notation	**Definition**
$t(\sigma)$	Period of signal σ
$d(\sigma)$	Deadline of signal σ
$p(\sigma)$	Size (in bytes) of signal σ
$\delta(\sigma)$	Domains of signal σ

Notation	**Definition**
$T(\gamma)$	Period of frame γ
$D(\gamma)$	Deadline of frame γ
$P(\gamma)$	Payload size (in bytes) of frame γ
$\Delta(\gamma)$	Set of domains of signals packed in frame γ
$\mathcal{S}(\gamma)$	Set of signals packed in frame γ
$C(\gamma)$	Worst-case transmission time (WCTT) of frame γ
$\pi(\gamma)$	Priority level of frame γ

Table 2.2: Parameters of each signal and frame

2.4 Overview of the Two-Stage Optimization Procedure with BCGM

The multi-domain frame packing problem for CAN-FD is NP-hard. This follows directly from the NP-hardness of the special case of single-domain frame packing problem [5]. To cope with this complexity, we consider an optimization procedure that consists of two stages, as illustrated in Figure 2.3. In the first stage, we try to find a signal-to-frame packing with minimum total bandwidth utilization. In the second stage, given the signal-to-frame packing, we perform priority assignment to all the frames such that the response time of each frame falls within its deadline. We choose to minimize the total bandwidth in the first stage for two reasons: one is that this is also the overall objective, the other is that smaller bus bandwidth utilization generally leads to better schedulability. However, the latter is not always the case. When a candidate frame packing produced by Stage 1 is not schedulable, all the frames or a subset thereof are repacked, this time focusing on improving schedulability. The two-stage procedure iterates until a feasible solution is found or after a certain number of iterations of repacking have been completed.

As discussed in subsequent sections, we present two instantiations of the optimization procedure. One instantiation formulates the problem in the first stage as an integer linear program (ILP) and leverages existing solvers to find an optimal solution (i.e., one with minimum total bandwidth utilization). In case of unschedulability, the second stage iteratively produces another packing by sacrificing optimality by a certain amount, with additional constraints in the ILP model. As will be seen in Section 2.5, the ILP formulation is somewhat intricate due to the discontinuous nature of the frame sizes available under CAN-FD. Due to the number of constraints in the ILP formulation, this approach does not scale to large systems. The second instantiation of the procedure in Figure 2.3 uses a fast heuristic in the first stage.

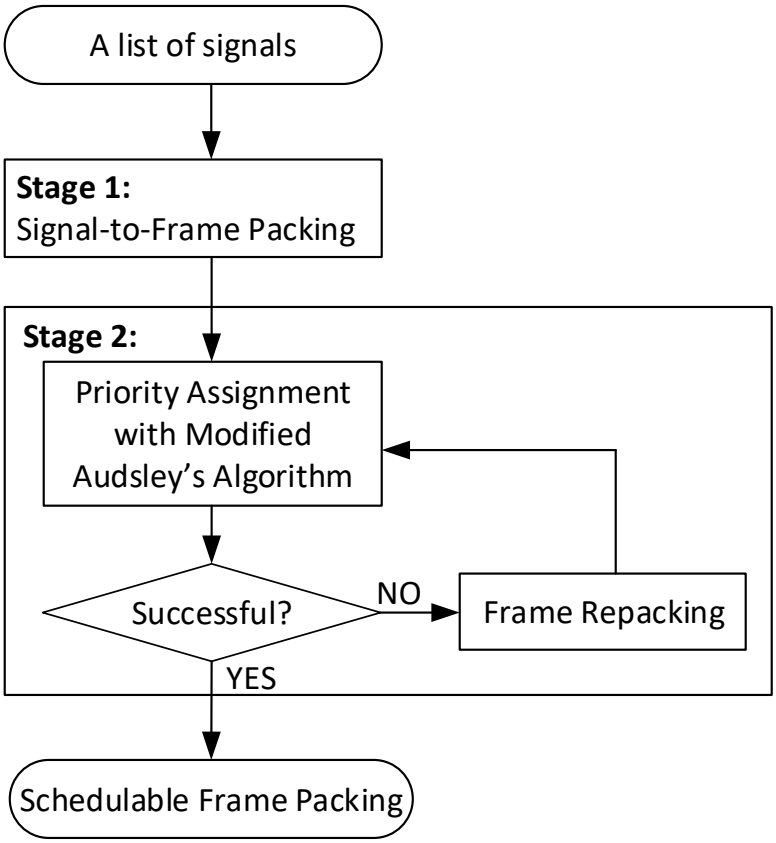


Figure 2.3: The two-stage optimization procedure.

Based on the observation that the first stage, namely the problem of signal-to-frame packing, is related to the bin packing problem, we develop a bandwidth-based best-fit (greedy) approach, where each signal is packed into a candidate frame that minimizes the total bandwidth utilization over all the domains. The second stage directly repacks a selected list of frames in case of unschedulability.

The problem of frame priority assignment can be solved efficiently. We propose a modified version of the Audsley's algorithm [2] that only needs to check a quadratic number of candidate priority assignments. Similar to Audsley's algorithm, it iteratively picks a frame that can be assigned a particular priority level starting from the lowest priority. However, when

choosing a candidate frame, it should guarantee that assigning the priority does not violate the schedulability in any domain to which the frame will be transmitted. Here, the priority order of frames remains the same across all domains, as the gateway is assumed to forward the frames without repacking or reassigning frame identifiers.

2.5 ILP-based Approach for Multi-Domain CAN-FD System with BCGM

2.5.1 ILP formulation for Signal-to-Frame Packing

Since each frame may only contain signals sent by the same ECU and we do not consider schedulability in the first stage, it is sufficient to perform frame packing for each ECU separately. Hence, we present the ILP formulation considering the set of signals $\mathcal{S}(\psi_{i,j})$ generated by ECU $\psi_{i,j}$. The total bandwidth utilization is the sum of the utilization values over all the ECUs.

We generate a set of virtual frames $\Gamma_{i,j} = \{\gamma_l \mid l = 1, ..., |\mathcal{S}(\psi_{i,j})|\}$, one for each signal in $\mathcal{S}(\psi_{i,j})$. Thus, $\Gamma_{i,j}$ consists of the maximum number of frames that could be used for packing the signals in $\mathcal{S}(\psi_{i,j})$; a packing may use only a subset of $\Gamma_{i,j}$. Each virtual frame $\gamma_l \in \Gamma_{i,j}$ is represented as a tuple $\langle T(\gamma_l), D(\gamma_l), P(\gamma_l)\rangle$, which specifies respectively the period, deadline and size of the frame. Here, we fix the period of each frame to be the period of its corresponding signal. Thus, $\forall l$, $T(\gamma_l) = t(\sigma_l)$. However, the deadline $D(\gamma_l)$ and size $P(\gamma_l)$ depend on the signals packed in γ_l. We use a binary *parameter* to denote whether a signal

shall be transmitted in a particular domain.

$$Y_{k,e} = \begin{cases} 1 & \text{if the destination of } \sigma_k \text{ is in domain } \Delta_e \\ 0 & \text{otherwise.} \end{cases}$$

Note that this information is available as part of the input. Thus, $Y_{k,e}$ is *not* a variable in the ILP formulation. We define a binary (decision) variable $x_{k,l}$ to indicate the mapping of signals to frames:

$$x_{k,l} = \begin{cases} 1 & \text{if signal } \sigma_k \text{ is packed into frame } \gamma_l \\ 0 & \text{otherwise.} \end{cases}$$

Each signal should be assigned to one and only one frame:

$$\forall k : \quad \sum_{l=1}^{|\mathcal{S}(\psi_{i,j})|} x_{k,l} = 1 \tag{2.7}$$

The period of a frame should be a divisor of the period of any signal assigned to the frame:

$$\forall k, l \text{ such that } t(\sigma_k) \bmod T(\gamma_l) \neq 0 : \quad x_{k,l} = 0 \tag{2.8}$$

If two signals have non-harmonic periods, they should not be packed in the same frame:

$$\forall k, m \text{ such that } t(\sigma_k) \geq t(\sigma_m) \bigwedge t(\sigma_k) \bmod t(\sigma_m) \neq 0, \ \forall \gamma_l : \ x_{k,l} + x_{m,l} \leq 1 \tag{2.9}$$

Since some frames may not be assigned any signals during the packing, we must ensure that they are not taken into account while computing the bandwidth utilization. To do this, we

use a binary variable ρ_l to indicate if γ_l contains any signals. Hence,

$$\frac{\sum_{k=1}^{|\mathcal{S}(\psi_{i,j})|} x_{k,l}}{|\mathcal{S}(\psi_{i,j})|} \leq \rho_l \bigwedge \rho_l \leq \sum_{k=1}^{|\mathcal{S}(\psi_{i,j})|} x_{k,l} \tag{2.10}$$

For each frame γ_l, we introduce a variable z_l that represents the sum of the sizes (in bytes) of the signals packed in γ_l:

$$z_l = \sum_{k=1}^{|\mathcal{S}(\psi_{i,j})|} x_{k,l} \cdot p(\sigma_k) \tag{2.11}$$

The maximum size of a frame is 64 bytes. Hence, the total size of the signals assigned to a frame should be no more than 64 bytes:

$$\forall l : \; z_l \leq 64 \tag{2.12}$$

As described in Section 1.1.1, CAN-FD allows 16 different frame payload sizes (0 through 8, 12, 16, 20, 24, 32, 48 and 64 bytes). Hence, the size $P(\gamma_l)$ of any frame γ_l can be modeled as a discontinuous function defined by

$$P(\gamma_l) = \begin{cases} z_l, & 0 \leq z_l \leq 8 \\ 12, & 8 < z_l \leq 12 \\ 16, & 12 < z_l \leq 16 \\ 20, & 16 < z_l \leq 20 \\ 24, & 20 < z_l \leq 24 \\ 32, & 24 < z_l \leq 32 \\ 48, & 32 < z_l \leq 48 \\ 64, & 48 < z_l \leq 64 \end{cases}$$

We define eight new binary variables λ_1, λ_2, ..., λ_8, which determine the ranges of the z_l variables.

$$\begin{cases} z_l \leq 8 + M(1-\lambda_1) & \\ z_l \leq 12 + M(1-\lambda_2) & \bigwedge\ z_l + M(1-\lambda_2) > 8 \\ z_l \leq 16 + M(1-\lambda_3) & \bigwedge\ z_l + M(1-\lambda_3) > 12 \\ z_l \leq 20 + M(1-\lambda_4) & \bigwedge\ z_l + M(1-\lambda_4) > 16 \\ z_l \leq 24 + M(1-\lambda_5) & \bigwedge\ z_l + M(1-\lambda_5) > 20 \\ z_l \leq 32 + M(1-\lambda_6) & \bigwedge\ z_l + M(1-\lambda_6) > 24 \\ z_l \leq 48 + M(1-\lambda_7) & \bigwedge\ z_l + M(1-\lambda_7) > 32 \\ z_l \leq 64 + M(1-\lambda_8) & \bigwedge\ z_l + M(1-\lambda_8) > 48 \end{cases} \tag{2.13}$$

where M is a large enough constant. Hence, the size of a frame can be expressed as

$$P(\gamma_l) = \lambda_1 \cdot z_l + 12\lambda_2 + 16\lambda_3 + 20\lambda_4 + 24\lambda_5 + 32\lambda_6 + 48\lambda_7 + 64\lambda_8 \tag{2.14}$$

However, there is a product term, namely $\lambda_1 \cdot z_l$, in Equation (2.14). This can be linearized by introducing a new variable v_l as follows.

$$v_l = \lambda_1 \cdot z_l \quad \Rightarrow \quad v_l \leq z_l + M(1-\lambda_1) \bigwedge z_l \leq v_l + M(1-\lambda_1) \bigwedge v_l \leq M \cdot z_l \tag{2.15}$$

Therefore, Equation (2.14) can be rewritten as the following linear constraint:

$$P(\gamma_l) = v_l + 12\lambda_2 + 16\lambda_3 + 20\lambda_4 + 24\lambda_5 + 32\lambda_6 + 48\lambda_7 + 64\lambda_8 \tag{2.16}$$

To calculate the WCTT of frame γ_l using Equation (1.2), we note that the ceiling function $\left\lceil \frac{P(\gamma_l)-16}{64} \right\rceil$ can only take on two values: 0 if $P(\gamma_l) \leq 16$ and 1 otherwise. Hence, we introduce

a binary variable u_l to represent it. The constraints on u_l are as follows:

$$P(\gamma_l) + M(1 - u_l) \;>\; 16 \bigwedge P(\gamma_l) \;\leq\; 16 + M \cdot u_l \tag{2.17}$$

Now, the expression for WCTT becomes

$$C(\gamma_l) \;=\; 32\, t_a + (28 + 5u_l + 10P(\gamma_l))\, t_d \tag{2.18}$$

The total bandwidth utilization for all the frames for an ECU $\psi_{i,j}$ in the CAN-FD network can be expressed as follows:

$$\sum_{l}^{|\mathcal{S}(\psi_{i,j})|} \left[\rho_l \cdot \frac{C(\gamma_l)}{T(\gamma_l)} + \sum_{e \neq i} \left(\eta_{l,e} \cdot \frac{C(\gamma_l)}{T(\gamma_l)}\right)\right] \tag{2.19}$$

where the first part $\rho_l \cdot \frac{C(\gamma_l)}{T(\gamma_l)}$ corresponds to the bandwidth utilization over the source domain Δ_i, and the second part $\sum_{e \neq i} \left(\eta_{l,e} \cdot \frac{C(\gamma_l)}{T(\gamma_l)}\right)$ corresponds to the bandwidth utilization over all the destination domains. In Equation (2.19), $\eta_{l,e}$ is a binary variable to determine whether the frame γ_l has any signal with a destination in domain Δ_e. Using the binary parameter $Y_{k,e}$ defined earlier, $\eta_{l,e}$ can be defined as

$$\eta_{l,e} = \begin{cases} 1 & \text{if } \sum_{k=1}^{|\mathcal{S}(\psi_{i,j})|}(x_{k,l} \cdot Y_{k,e}) \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

The linear constraints that enforce the definition of $\eta_{l,e}$ are as follows:

$$\frac{\sum_{k=1}^{|\mathcal{S}(\psi_{i,j})|}(x_{k,l} \cdot Y_{k,e})}{|\mathcal{S}(\psi_{i,j})|} \;\leq\; \eta_{l,e} \;\leq\; 1 \bigwedge \eta_{l,e} \;\leq\; \sum_{k=1}^{|\mathcal{S}(\psi_{i,j})|} (x_{k,l} \cdot Y_{k,e}) \tag{2.20}$$

Algorithm 1 Modified Audsley's Algorithm for Multi-Domain CAN-FD

```
1: procedure AudsleyMultiDomain (Γ)
2:     Let N = |Γ|, Create a list Q containing all the frames in Γ
3:     for π = N downto 1 do
4:         for each frame γ ∈ Q do
5:             flag_found = FALSE
6:             for each Δ_i ∈ Δ(γ) do
7:                 R(γ) = ComputeResponseTime (γ, Δ_i, Γ)
8:             if γ is schedulable in all Δ_i ∈ Δ(γ) then
9:                 Assign priority level π to γ
10:                Remove γ from Q
11:                flag_found = TRUE
12:                break
13:        if flag_found is FALSE then
14:            Report unschedulability and return
15:    Report schedulability
```

The objective is to minimize the total bandwidth utilization of all the frames:

$$\min \sum_{i}^{|\mathbb{D}|} \sum_{j}^{n_i} \sum_{l}^{|\mathcal{S}(\psi_{i,j})|} \left[\rho_l \cdot \frac{C(\gamma_l)}{T(\gamma_l)} + \sum_{e \neq i} \left(\eta_{l,e} \cdot \frac{C(\gamma_l)}{T(\gamma_l)} \right) \right] \tag{2.21}$$

In Equation (2.21), $\rho_l \cdot \frac{C(\gamma_l)}{T(\gamma_l)}$ and $\eta_{l,e} \cdot \frac{C(\gamma_l)}{T(\gamma_l)}$ are both a product of a binary variable and a real variable. They can be linearized in a manner similar to that of Equation (2.15).

2.5.2 Modified Audsley's Algorithm for Priority Assignment

In order to assign priority identifiers to all the frames, we extend Audsley's algorithm [2] to the multi-domain case (Algorithm 1). The input to the algorithm is the set of frames Γ for all the domains. Similar to Audsley's algorithm, priority levels are assigned iteratively to all the frames starting from lowest to highest (Lines 3–15). At each iteration, a priority level is assigned to the first frame γ that satisfies the schedulability constraints over all the domains belonging to $\Delta(\gamma)$ (Lines 8–12). If a priority level cannot be assigned to any of the frames

(i.e., flag_found is FALSE), the algorithm reports unschedulability (Lines 13–14). If all the frames are assigned a unique priority, the algorithm is successful in finding a schedulable priority assignment.

Using the approach in [11], it can be easily shown that the schedulability of a multi-domain frame (i.e., whether it meets the deadline requirement in all its domains) satisfies all the three conditions which are necessary and sufficient to provide an optimal priority assignment. Hence, our extension to Audsley's algorithm for the multi-domain CAN-FD system is optimal for finding a schedulable priority assignment.

2.5.3 Handling Infeasibility

Although in principle our frame packing scheme supports schedulability (since it minimizes bandwidth utilization which indirectly helps to reduce network traffic), there are cases when the modified Audsley's algorithm returns infeasibility.

In the case of ILP, when we encounter infeasibility, we call the solver again after relaxing the optimal value of the objective function (by doubling the optimality gap in each iteration) and setting a time limit of one hour for each iteration. We report infeasibility if no feasible priority assignment is found even after a given number of iterations.

2.6 Greedy Algorithm-based Approach for Multi-Domain CAN-FD System with BCGM

The ILP approach discussed in the previous section provides an optimal packing of the signals into frames with respect to bandwidth utilization. However, due to its exponential

time complexity, it does not scale well to large sets of signals. Therefore, we propose a greedy heuristic (Algorithm 2) for the frame packing step. The heart of the algorithm presents the steps for packing the signals from one ECU; the outermost loop ensures that the steps are iterated over all the ECUs.

2.6.1 Description of the Heuristic for Signal-to-Frame Packing

The algorithm first sorts the input signals for each ECU (Line 3 in Algorithm 2) on the basis of a parameter such as the period, size or the input bandwidth utilization (which is given by the size/period) of signals. It then uses a Bandwidth Best-Fit approach to pack the signals into frames as follows. Starting from the first signal, each signal is placed in a frame that minimizes the total bandwidth utilization of the system (*over all the domains*). The steps shown in lines 6 and 7 create a new frame and add the signal to it. To obtain the bandwidth utilization for a frame, we use Equation (2.6) to compute the utilization over each of its destination domains and then take their sum. The total bandwidth utilization of the system is the sum of the bandwidth utilization over all the frames (Equation (2.19)). Further, lines 8–14 compute and store the total bandwidth utilization by temporarily adding the signal to an existing frame. Before a signal is assigned to an existing frame, the 'if(σ_k can be added to F_j)' condition (Line 9 in Algorithm 2) checks (i) whether the frame can accommodate the new signal (i.e., the total size of all the signals in the frame is at most 64); and (ii) whether the period of the new signal is harmonic with the periods of the other signals in that frame. The steps in lines 15 and 16 decide whether it is beneficial to add the current signal to a new frame or to one of the existing frames. The output of the algorithm is a list of frames (Γ) which stores the frames created in each step.

The quality of the packing depends on the sorting criterion used in Line 3. In our experi-

Algorithm 2 Greedy Algorithm

```
1: procedure Greedy-Bw-Best-Fit (Ψ, S)
2:     for each ECU ψ_i ∈ Ψ do
3:         Sort(S(ψ_i))
4:         Number of frames n = 0, list of frames Γ = ∅
5:         for each signal σ_k in S(ψ_i) do
6:             Create a new frame F_{n+1} containing only σ_k
7:             Compute the total BW utilization u_{n+1} of frames F_1, ..., F_n, F_{n+1}
8:             for j = 1 to n do
9:                 if (σ_k can be added to F_j) then
10:                    Add σ_k to F_j
11:                    Compute the total BW utilization u_j of frames F_1, ..., F_n
12:                    Remove σ_k from F_j
13:                else
14:                    Set u_j to infinity
15:            Find the smallest u_j among u_1, ..., u_{n+1} and pack σ_k in F_j
16:            if (j == n + 1) then add F_{n+1} to Γ and set n = n + 1
17:    Return Γ
```

ments, we compare different sorting methods using each of the above parameters (i.e., period, size and size/period) in both increasing and decreasing orders.

Time Complexity Analysis: For each ECU ψ_i, we show that the above heuristic runs in $O(s^2 f)$ time, where $s = |S_i|$ is the number of signals for the ECU and f is the number of frames in the resulting packing. To begin with, sorting the set $\mathcal{S}(\psi_i)$ can be done in $O(s \log s)$ time. Now, for each signal $\sigma \in \mathcal{S}(\psi_i)$, the time for finding the best placement into a frame can be estimated as follows. Let $\Gamma(\psi_i) = \{\gamma_1, \gamma_2, \ldots, \gamma_r\}$ denote the current set of frames when σ is considered. Creating a new frame containing just σ can be done in $O(1)$ time. As mentioned above, testing whether σ can be added to a frame γ_i involves two checks involving the size of the frame and the harmonicity of periods of the signals currently in the frame. It is easy to see that each of these checks can be done in time $O(|\gamma_i|)$, where $|\gamma_i|$ denotes the number of signals in γ_i. Thus, the total time for checking whether σ can be added to each of the existing frames is $O(\sum_{i=1}^{r} |\gamma_i|) = O(s)$, since all the frames together contain at

most s signals. For each placement of σ in a frame, it is also easy to see that computing the bandwidth utilization (as explained in the description of the heuristic) for signal σ can be done in $O(r + \sum_{i=1}^{r} |\gamma_i|) = O(s)$ time since $r \leq f \leq s$ and as observed earlier, $\sum_{i=1}^{r} |\gamma_i| \leq s$. As we need to compute the bandwidth utilization for at most $f + 1$ alternatives (including the new frame containing only σ), the time used for this step is $O(sf)$. In other words, for each signal, the greedy heuristic uses $O(sf)$ time. So, over all the s signals in $\mathcal{S}(\psi_i)$, the time complexity of the heuristic is $O(s^2 f)$.

2.6.2 Handling Infeasibility

In the second stage of the optimization, in case Algorithm 1 (i.e., the modified Audsley's Algorithm) fails to find a schedulable priority assignment, we propose a repacking method, with three variations, so that the frames may become schedulable. Our repacking strategy consists of two parts: the first part unpacks a selected set of frames based on certain conditions, and the second part repacks the signals removed from frames. The first part (unpacking of frames) is based on the following two methods.

1. **Unpacking Based on Destination Domains:** In case of a multi-domain system, Algorithm 1 reports infeasibility when a particular priority level cannot be assigned to any frame. This infeasibility could occur due to certain domains. Therefore, our first unpacking method attempts to separate the signals destined for different domains. The intuition behind such an unpacking can be understood from the following example. Consider a frame with 14 signals: σ_1, σ_2, ... σ_{14}. Suppose the first 12 signals have a total size of 40 bytes and their destination domain is D_1 while signals σ_{13} and σ_{14} have a total size of 2 bytes and their destination domain is D_2. Thus, such a frame carries an extra payload of 40 bytes to domain D_2. It could be beneficial to remove

the signals intended for domain D_2 from the frame so that the overall schedulability (in particular for D_2) may be improved.

2. **Unpacking Based on Deadline:** Increasing the deadline of a frame is another method we adopt in order to satisfy the schedulability constraints. We apply this in our unpacking scheme by separating the signals with the smallest deadline in a frame, thereby increasing the frame's deadline and its schedulability.

We apply at most one unpacking scheme per frame in an iteration. If the first criterion (destination domain based unpacking) is applicable to a frame, then we unpack the corresponding signals and do no further unpacking for this frame. If the first criterion does not apply to a frame (i.e., all the signals in the frame have the same destination domain), then we check the second criterion (deadline based unpacking). If neither of the criteria is met for a frame, we do not unpack that frame. After unpacking the signals, we repack them into existing frames using first-fit, best-fit and worst-fit heuristics. The repacking should satisfy the previously stated constraints of the problem (i.e., a frame should only have signals from the same ECU, all the signals in a frame should have harmonic periods and the total size of the signals in a frame should not exceed 64 bytes). The repacking step may suitably expand or shrink the size of a frame to handle the addition and removal of signals respectively.

We consider three different sets of frames as candidates for the unpacking and repacking steps. For each signal in the set, we unpack signals based on the above criteria and then repack them into existing frames (or generate new frames) using first-fit (FF), best-fit (BF) and worst-fit (WF) methods.

1. *All the frames*: In this case, we consider all the frames. We refer to this variation as the 'All' heuristic.

2. *Unassigned frames from Audsley's method:* Here, we unpack only those frames for

which Algorithm 1 could not assign a priority level. That is, at the priority level where Algorithm 1 fails to find a schedulable frame from the remaining set of frames, we consider this set for unpacking. We call this variation the 'Unassigned' heuristic.

3. *Irreducible subset:* The idea behind computing an irreducible subset is similar to the computation of a minimal unsatisfiable core of a Boolean formula in conjunctive normal form [45]. When Algorithm 1 reports infeasibility, we compute a set of frames Γ', called an **irreducible subset**, satisfying the following condition: the set Γ' does not satisfy the schedulability constraints, but for any frame $\gamma \in \Gamma'$, the set $\Gamma' - \{\gamma\}$ becomes schedulable. We only unpack the frames which form an irreducible subset, based on the above two criteria. We refer this variation as 'Irreducible subset' heuristic.

The reason for developing three variations of the repacking strategy is due to the complexity of making a given set of frames schedulable over the network. Since schedulability depends on a number of factors such as deadlines, traffic congestion over the network, sizes of frames, etc., there is no single factor which can be manipulated in order to obtain schedulability. The three different methods of repacking target different input sets. For example, for a particular input, it might be beneficial to unpack and repack a smaller set of 'problematic' frames whereas another input might require a larger set of frames to be repacked. In the former case, the 'Irreducible subset' approach could be more beneficial, and in latter case, the 'All' approach might yield better results.

2.7 Heuristics for Multi-Domain CAN-FD System with ACGM

An important reason for the increase in bandwidth utilization in a multi-domain system is the replication of signals over domains that are not their intended destinations. This happens because the basic gateway just forwards the received frames; it does not decompose the signals in a frame based on their destination domains.

Hence we propose an advanced central gateway model (ACGM) as described in Section 2.3 (Figure 2.2). By using the ACGM, we intend to further reduce the bandwidth utilization in multi-domain CAN-FD systems. Therefore, in addition to the signal-to-frame packing at the ECU level, we also consider the decomposition of each frame at the central gateway. Both the ECU-level packing and the decomposition at the ACGM are statically determined during the design phase.

We saw earlier that the frame packing problem for the multi-domain CAN-FD system is a very complex one, and an additional level of complexity is introduced by the advanced gateway model. To exploit the decomposing ability of the ACGM, we propose an extension to the previous signal-to-frame packing framework for multi-domain CAN-FD systems connected by an ACGM; this extension is shown in Figure 2.4. Similar to the framework described in Section 2.4, Figure 2.4 presents a two stage packing framework. The first stage consists of two steps 1) a signal-to-frame packing for each ECU and 2) frame decomposition (of each frame obtained in step (1)) at the ACGM based on the destination domains of its constituent signals (as described in Figure 2.2). In the second stage we assign priorities to the packed frames using Audsley's algorithm such that each frame has a response time within its deadline. However if such a priority assignment is not feasible then we can follow the same procedure of repacking the frames (for the purpose of achieving schedulability) as

described in Section 2.6.2.

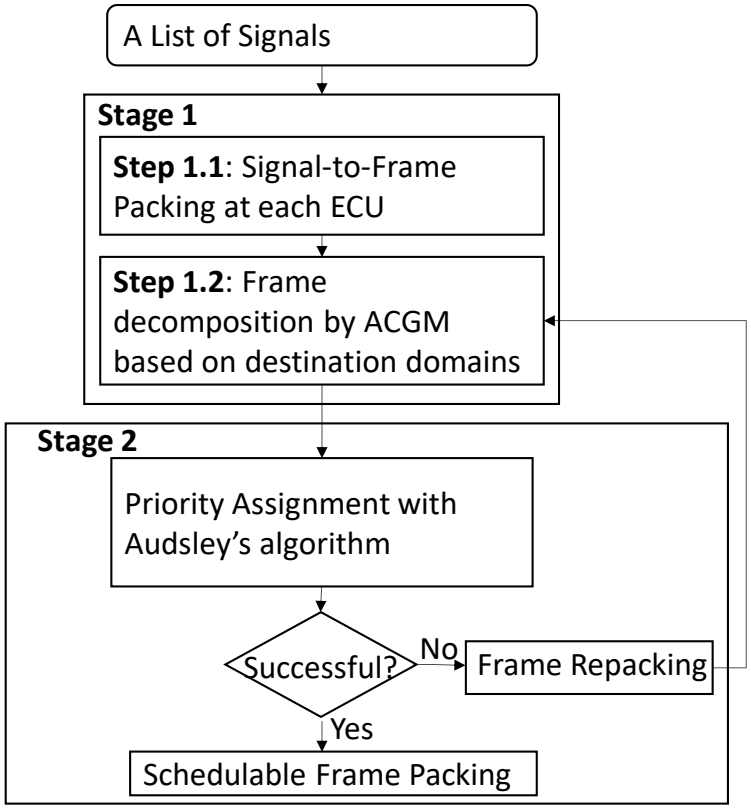


Figure 2.4: Proposed Signal-to-Frame packing and repacking for ACGM.

The proposed ACGM is a simple addition to the multi-domain CAN-FD system and it has the potential to significantly decrease the bandwidth utilization and enhance schedulability. Some challenges that must be overcome in using this advanced gateway model in the automotive industry are presented in Section 2.7.2.

We now prove formally that the decomposition step by an ACGM does not cause any increase in bandwidth utilization. Our experimental results (Section 2.8.2) show the decomposition step by an ACGM can indeed decrease the bandwidth utilization significantly.

Theorem 2.1. *The total bandwidth utilization of a frame (over all its destination domains)*

after decomposition by an ACGM is always less than or equal to its total bandwidth utilization prior to decomposition.

Proof: Consider a frame γ which has signals for n destination domains. We first compute the total bandwidth utilization of transmitting γ over n destination domains when it is not decomposed at the central gateway. If $P(\gamma)$ is the payload and $T(\gamma)$ is the period of frame γ, then the bandwidth utilization over one domain is given by $\frac{C(\gamma)}{T(\gamma)}$, where $C(\gamma)$ can be obtained from Equation (1.2) by replacing p with $P(\gamma)$. Thus, the total bandwidth utilization is given by the utilization over the source domain and over the destination domains:

$$\frac{C(\gamma)}{T(\gamma)} + n * \frac{C(\gamma)}{T(\gamma)} = (n+1) * \frac{C(\gamma)}{T(\gamma)} \tag{2.22}$$

We now compute the total bandwidth utilization of transmitting the frame γ when it is decomposed at the gateway. Since γ has signals for n destination domains, n new frames are created after the decomposition step. Let $\gamma_1, \gamma_2, ...\gamma_n$ denote the new frames and let $P(\gamma_1), P(\gamma_2), ..., P(\gamma_n)$ denote their respective payloads. Since each new frame packs a subset of the signals from the original frame, we have $P(\gamma_i) \leq P(\gamma),\ \ 1 \leq i \leq n$.

The total utilization in this case is given by

$$\frac{C(\gamma)}{T(\gamma)} + \sum_{i=1}^{n} \frac{C(\gamma_i)}{T(\gamma_i)} \tag{2.23}$$

Here, the first term denotes the bandwidth utilization of γ on the source domains and the second part denotes the bandwidth utilization of the decomposed frames on the respective destination domains.

Claim 1: Every new frame $\gamma_i,\ 1 \leq i \leq n$ created from a decomposition of frame γ has a worst case transmission time (WCTT) less than or equal to that of the original frame; i.e.

$C(\gamma_i) \leq C(\gamma),\ \ 1 \leq i \leq n.$

Proof of Claim 1: The WCTT function is a continuous non-decreasing function of payload $P(\gamma)$ of a frame γ, given by Equation (1.2). Since $P(\gamma_i) \leq P(\gamma)$, $1 \leq i \leq n$, Claim 1 follows.

Claim 2: $T(\gamma) \leq T(\gamma_i)$, $1 \leq i \leq n$.

Proof of Claim 2: This is a simple consequence of the fact that the period of a frame is the smallest period of the signals in the frame.

From Claims 1 and 2, it follows that the total utilization after decomposition given by Equation (2.23) is at most $(n+1)\,C(\gamma)/T(\gamma)$, the bandwidth utilization when there is no decomposition at the gateway (Equation (2.22)). This completes the proof of Theorem 2.1.

■

We now discuss four heuristics for decomposing frames at the ACGM.

1. **Greedy Heuristic with Decomposition at ACGM (Greedy-ACGM):** In this heuristic, the signal-to-frame packing described in Section 2.6.1 is used as the first step, where the decision to assign a signal to the available frames is based on the total bandwidth utilization computed over all the domains (using Equation (2.19)). In the second step, each packed frame is then decomposed as described in Algorithm 3. The intuition behind this method is to see the additional benefit of decomposition by an ACGM over our originally proposed multi-domain frame packing heuristic for CAN-FD.

2. **Greedy Heuristic with No-Cross-Domain-Consideration and with Decomposition at ACGM (Greedy-No-Cross-ACGM):** In this heuristic, the first step is the baseline approach as described in Section 2.8.1. As discussed earlier, the baseline approach takes into account only the first part of Equation (2.19) (the source domain

bandwidth utilization) but not the second part (cross-domain bandwidth utilization). Hence, the signal-to-frame packing decision is made on the basis of the bandwidth utilization of the frames on the source domain only (i.e., cross-domain bandwidth utilization is not considered). The decomposition step is similar to 1.

The idea behind using the baseline approach as the first step is to be less stringent than 1 in the original packing in order to reap more benefit from the decomposition step in terms of total bandwidth utilization.

3. **Greedy Heuristic with Decomposition-Consideration and with Decomposition at ACGM**

   (Greedy-Decompose-Consideration-ACGM): In this method, the first step of signal-to-frame packing uses the greedy algorithm described in Section 2.6.1; however, the difference here is that it computes bandwidth utilization of the frames using Equation (2.23) instead of Equation (2.19). Thus, the decision to pack a signal into one of the available frames or a new frame explicitly takes into account the subsequent decomposition at the central gateway. The second step is similar to that of 1.

4. **Greedy Heuristic with No Constraints and Decomposition at ACGM (Greedy-No-Constraint-ACGM):** In this heuristic, the first step signal-to-frame packing is similar to that of 2; however, in this method, in order to further relax the packing in Step 1, we remove the harmonicity constraint defined in Section 2.3. As stated above, this is done with the intention of reducing bandwidth utilization over the source domain transmission. The second step of decomposition is again similar to the one used in 1.

Algorithm 3 Decomposition of Frame at the Central Gateway

1: **procedure** DECOMPOSEFRAMEATGATWAY (γ)
2: Let $\delta_1, \delta_2, \ldots, \delta_k$ denote the distinct destination domains of the signals in γ (not including the source domain of γ).
3: **for** each destination domain $\delta_i, 1 \leq i \leq k$ **do**
4: create a new frame γ_i
5: **for** each $\sigma_j \in S(\gamma)$ **do**
6: **if** destination domain of σ_j is δ_x **then**
7: Add signal σ_j to γ_x
8: Set destination domain of γ_x as δ_x
9: Update Γ with the new frames

2.7.1 Schedulability Analysis

After obtaining a signal-to-frame packing from our proposed decomposition heuristics, we conduct a schedulability analysis as discussed in Section 1.1.1 using the CAN analysis in [10]. In case of an ACGM, each frame is transmitted on only one domain. This is due to the fact that when a frame is decomposed at the gateway, the new frames have identifiers which are different from that of the original frame. Therefore, to do a worst case analysis, we only need to make sure that all the frames on each domain are able to meet their deadlines.

We use Audsley's algorithm for assigning unique priority identifiers to all the frames. If such an assignment which also ensures that each frame meets its deadline is possible, then the set of frames is said to be schedulable; otherwise, the set of frames is unschedulable. Note that this is a pessimistic approach to ensuring schedulability. In case the set of frames are found to be unchedulable we can apply the procedure of repacking of frame as described in Section 2.6.2 in order to improve schedulability.

2.7.2 Application of ACGM in industry

In order to use our proposed model of a advanced central gateway in industry, the following challenges must be addressed.

1. Update the signal-to-frame packing look up table: The frame packing is determined statically and stored as a look up table for the receivers. In order to incorporate the decomposition step by an advanced gateway, the look up tables must be updated with the new frames and their respective destinations.

2. Limited number of frame identifiers: CAN/CAN-FD systems have a limited number of identifiers that can be assigned to the frames. The current practice uses 11-bit identifiers for frames, thus providing for a total of $2^{11} = 2048$ frames. The new frames created by the advanced gateway must ensure that this limit is not exceeded.

2.8 Experimental Results

2.8.1 Multi-Domain CAN-FD System with BCGM

In this section, we present a detailed evaluation of the proposed algorithms using synthetic systems. For these experiments, we generated synthetic systems according to the guidelines on real-world automotive benchmarks [39], with minor modifications. Specifically, we redistributed the share of signals with size larger than 64 bytes to the bin "33-64 bytes" (as for this work we only consider signals with size up to 64 bytes), and the share of signals sent by engine control tasks to those with periods between 1 and 20ms (as we do not consider the signals with angle-synchronous periods). Table 3.1 summarizes the distribution of signal periods and payload sizes used for generating the synthetic systems. Each signal is randomly

Period (ms)	**Share**	**Size** (Bytes)	**Share**
1	4%	1	35%
2	3%	2	49%
5	3%	4	13%
10	31%	5-8	0.8%
20	31%	9-16	1.3%
50	3%	17-32	0.5%
100	20%	33-64	0.4%
200	1%		
1000	4%		

Table 2.3: Signal parameters and their distribution

assigned a source and a destination domain (from the set of domain IDs) with a probability of $1/|\mathbb{D}|$, where $|\mathbb{D}|$ is the total number of domains. Therefore, the probability of a signal being cross-domain (i.e., the probability that its source and destination domains are different) is $1 - 1/|\mathbb{D}|$. In all our experiments, we use a system with three domains, 10 ECUs (in total) and vary the number of signals from 80 to 220. Hence for our experiments, the probability of a signal being cross-domain is 2/3.

Our experiments are conducted using a high performance computing cluster at Virginia Tech. This cluster has 4 x E7-8867v4 2.4 GHz (Broadwell) processors and 3TB, 2400MHz memory on a Unix platform. We used IBM's CPLEX as the ILP solver and implemented the algorithms in C++.

Comparison of Greedy Packing Heuristics

For all of our experiments in this section, we generated 5000 benchmarks for each system size (in terms of the number of signals).

We first evaluate the different greedy packing heuristic approaches by comparing the bandwidth utilization they provide after packing. We note that our greedy approach (Algorithm 2)

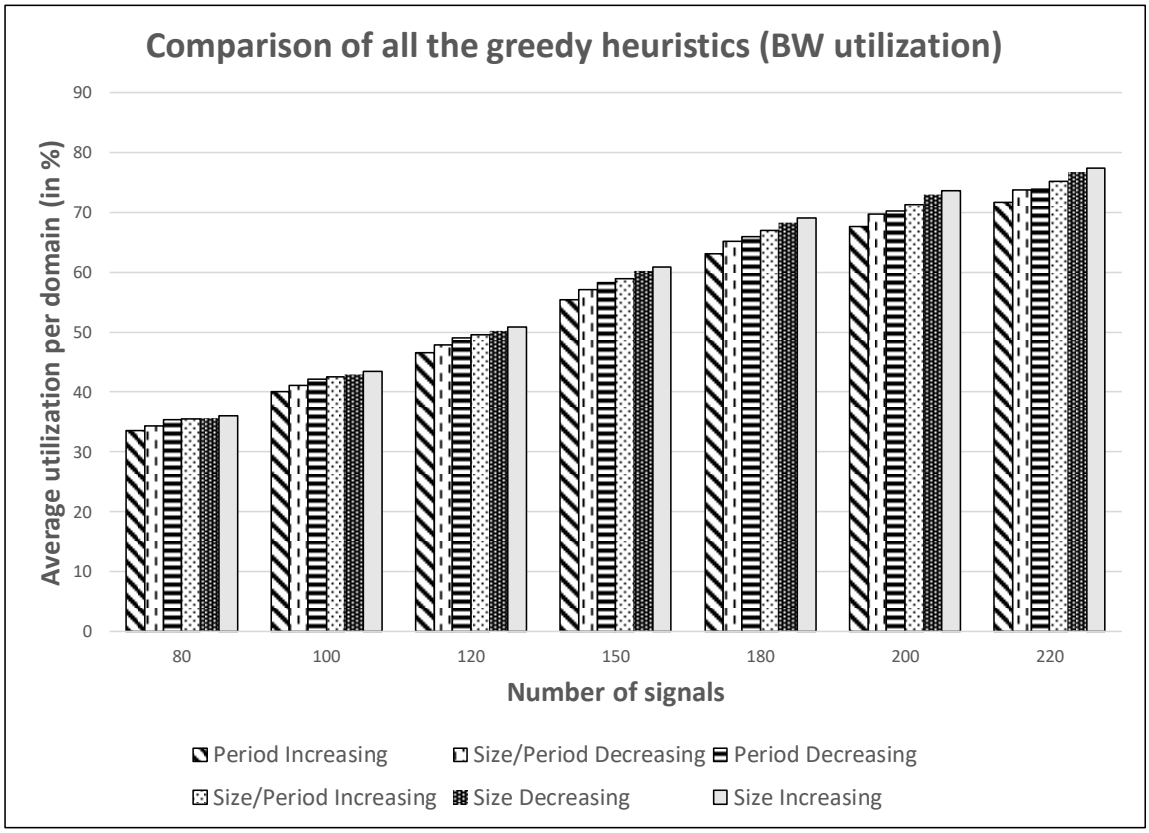


Figure 2.5: Comparison of the bandwidth utilization of all the proposed greedy heuristics.

leads to different bandwidth utilization values depending on the sorting criterion. We used the following parameters: period, size, and size/period (with increasing and decreasing orders). After the sorting step, each algorithm packs the signals in a greedy manner to optimize the bandwidth utilization.

Figure 2.5 shows the bandwidth utilization per domain of the greedy heuristics, where the utilization is averaged over all those systems which are schedulable. As seen from the figure, there is small but noticeable variation in bandwidth utilization among the different heuristics. However, in all cases, sorting by increasing period performs the best. This is consistent with the observation that it is beneficial to pack signals with similar periods together, which in

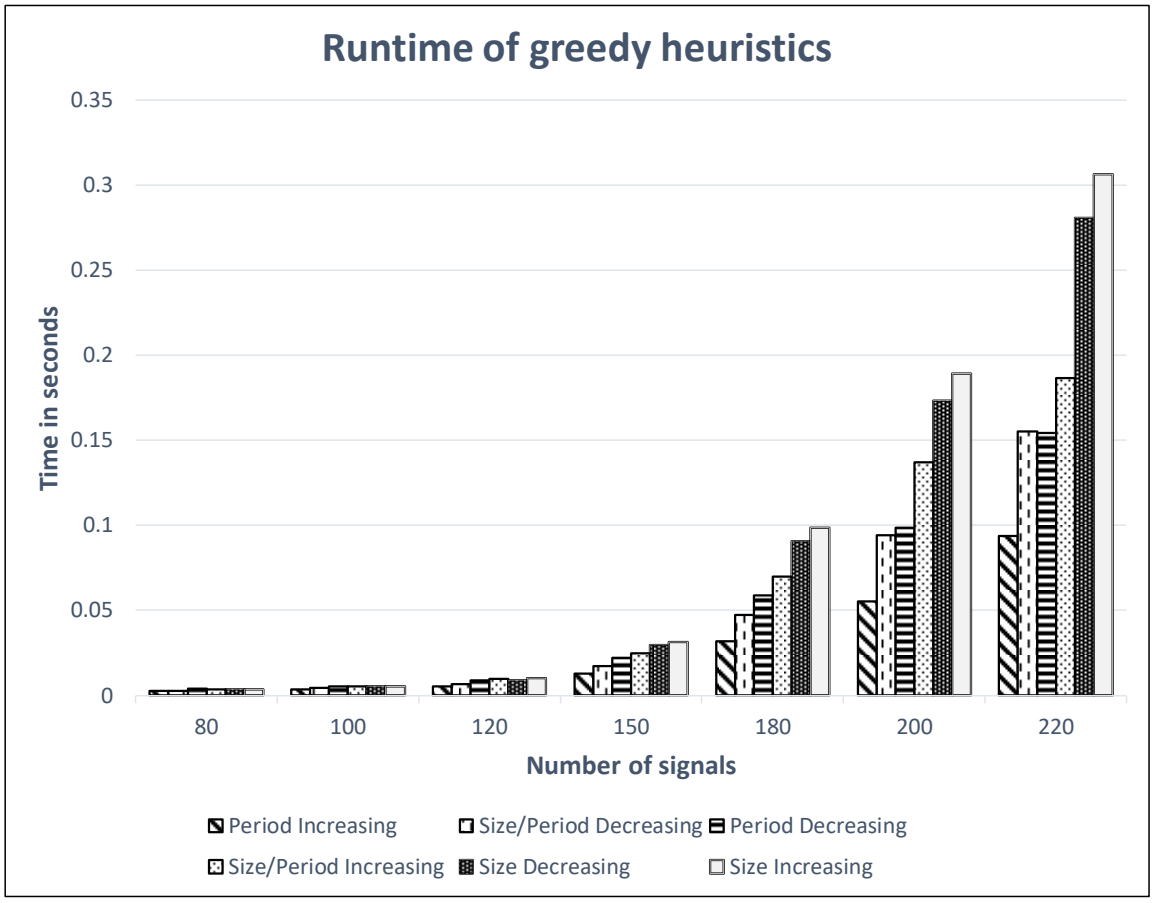


Figure 2.6: Comparison of the runtime for all the proposed greedy heuristics.

general reduces the total bandwidth utilization.

We also plot the runtime of the heuristics for each system size in Figure 2.6. It is clear from this figure that the heuristic of sorting the signals in increasing order of periods has the smallest runtime; that is, it has an advantage over the other sorting approaches in terms of algorithmic efficiency as well. This is due to the fact that (as pointed out in the time complexity analysis for the greedy approaches) the runtime of the greedy algorithm is a function of the number of frames created, and the algorithm that sorts the signals in increasing order of periods creates the least number of frames for each signal size (as

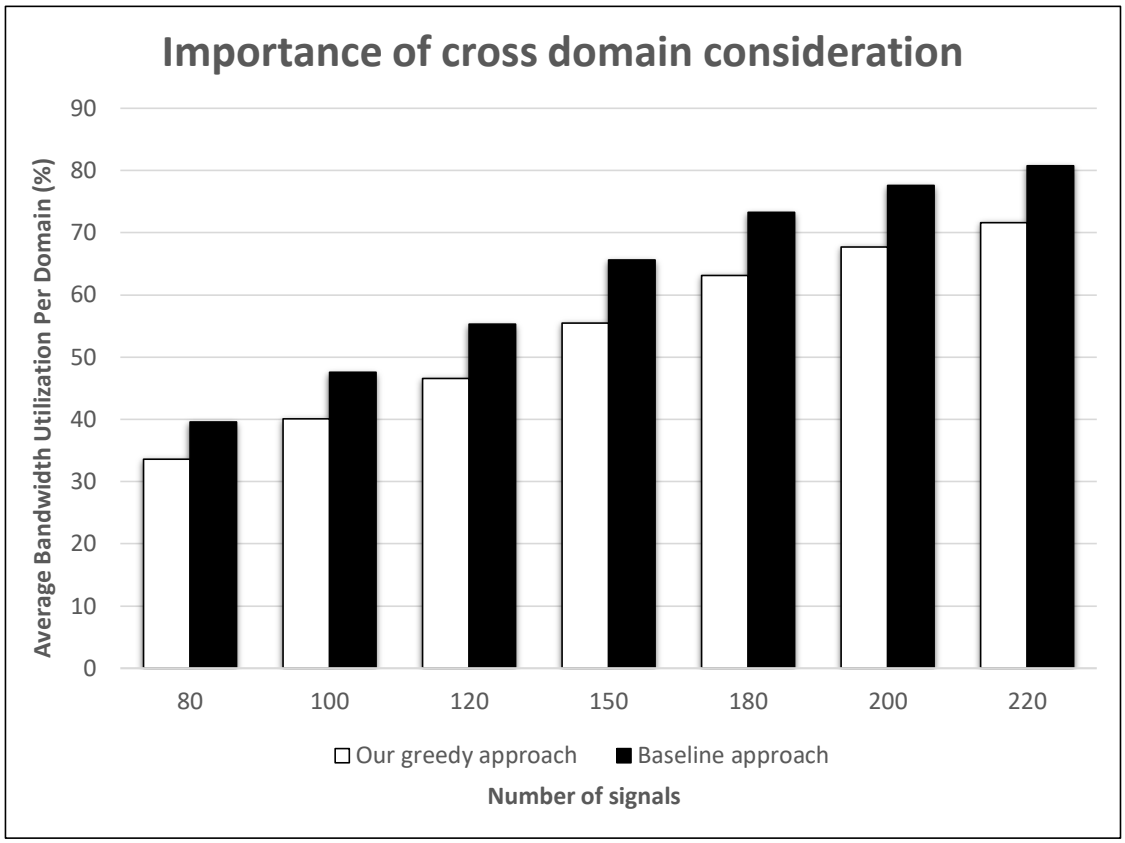


Figure 2.7: Comparing the greedy heuristic with a baseline packing approach which does not consider cross domain bandwidth while packing.

compared to the other heuristics). Hence, for the rest of the experiments, we use the heuristic that sorts the signals in increasing order of periods to represent all the greedy heuristics and compare it with the ILP-based approach.

Importance of Cross-Domain Consideration

In this experiment, we demonstrate the importance of considering the cross-domain utilization during packing of the signals. Since there is no existing work for multi-domain frame

packing in CAN-FD, we compare our greedy heuristic to a baseline approach which does not consider cross domain bandwidth utilization. Specifically, the baseline approach is the same as the greedy heuristic, except that it takes into account only the first part of Equation (2.19) (the source domain bandwidth utilization) but not the second part (cross-domain bandwidth utilization). For each system size, we tried 5000 benchmarks, and the bandwidth utilization represented is the average per domain over the 5000 benchmarks.

Figure 2.7 shows the significance of taking into account the cross-domain utilization for packing. We observe that there is a considerable reduction in bandwidth utilization per domain when frames are packed using our approach as opposed to the baseline approach: the typical reduction is in the range of 6% to 10% for utilization per domain. Also, the gap becomes larger as the number of signals increases.

Comparison of the Greedy Heuristic with ILP

In this experiment, we compare the ILP-based approach and the greedy heuristic in terms of their bandwidth utilization and the runtime. For these experiments, we used 100 synthetic systems for each size due to the excessively long runtime of ILP. We stopped at systems with 220 signals as ILP cannot scale to any larger systems: for systems with 4 domains, 15 ECUs and 250 signals, each of them takes about 7 hours on average.

Figure 2.8 illustrates the average bandwidth utilization per domain over the systems that are schedulable (either in the first packing attempt or after iteration/repacking). As can be observed from the figure, the bandwidth utilization of the greedy approach is quite close to that of ILP. The maximum mean difference on bandwidth utilization per domain is about 2.7% for systems with 220 signals.

Since Figure 2.8 gives only the average bandwidth utilization over all the systems, to better

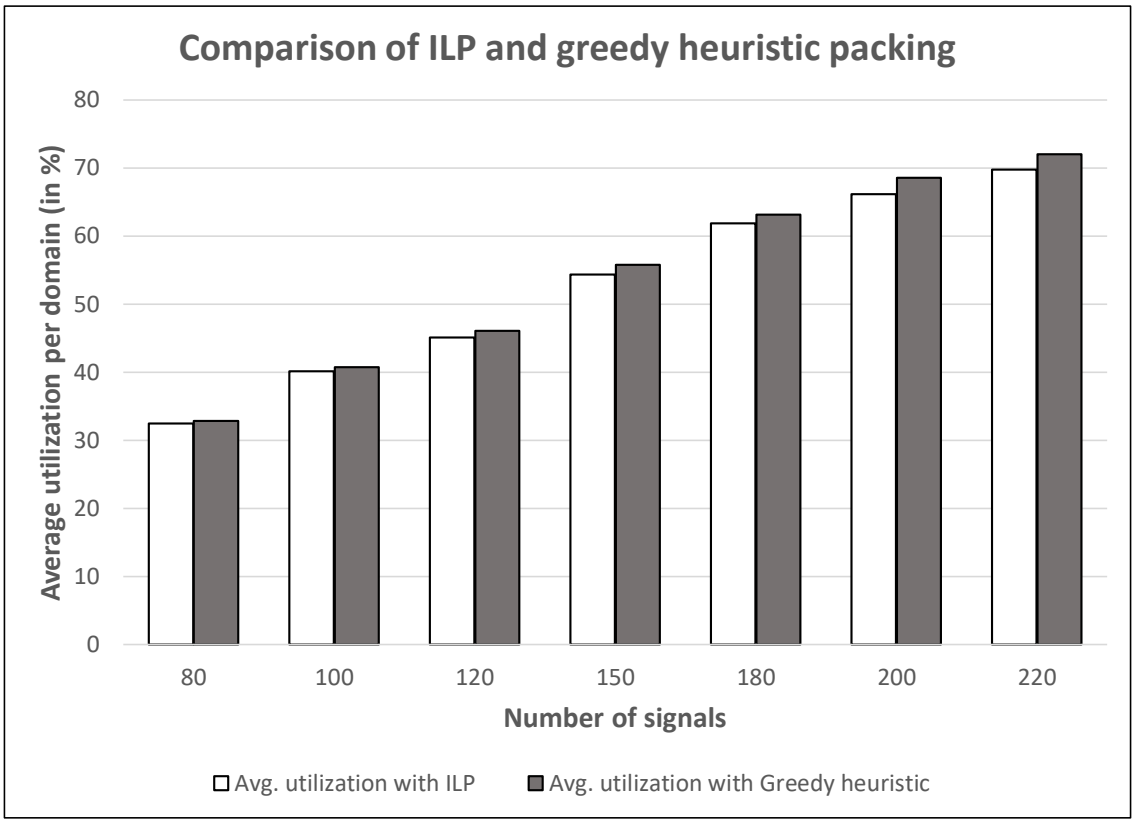


Figure 2.8: Greedy vs. ILP: bandwidth utilization per domain

compare the ILP and the greedy approaches, we present the variability of the difference between the utilization values reported by them in Figure 2.9. The gray bars represent the mean difference in percentage (over the 100 random systems, which are schedulable) between the bandwidth utilization (per domain) given by the ILP and the greedy approaches. The error bars represent the standard deviation of the difference. Figure 2.10 presents the distribution of the number of systems (that are schedulable) into different bins which represent the range of the difference between the average bandwidth utilization of the ILP and greedy approaches (as indicated in the legend). As the number of signals increases, the number of systems assigned to the larger bins also increases. Thus, Figures 2.9 and 2.10 show

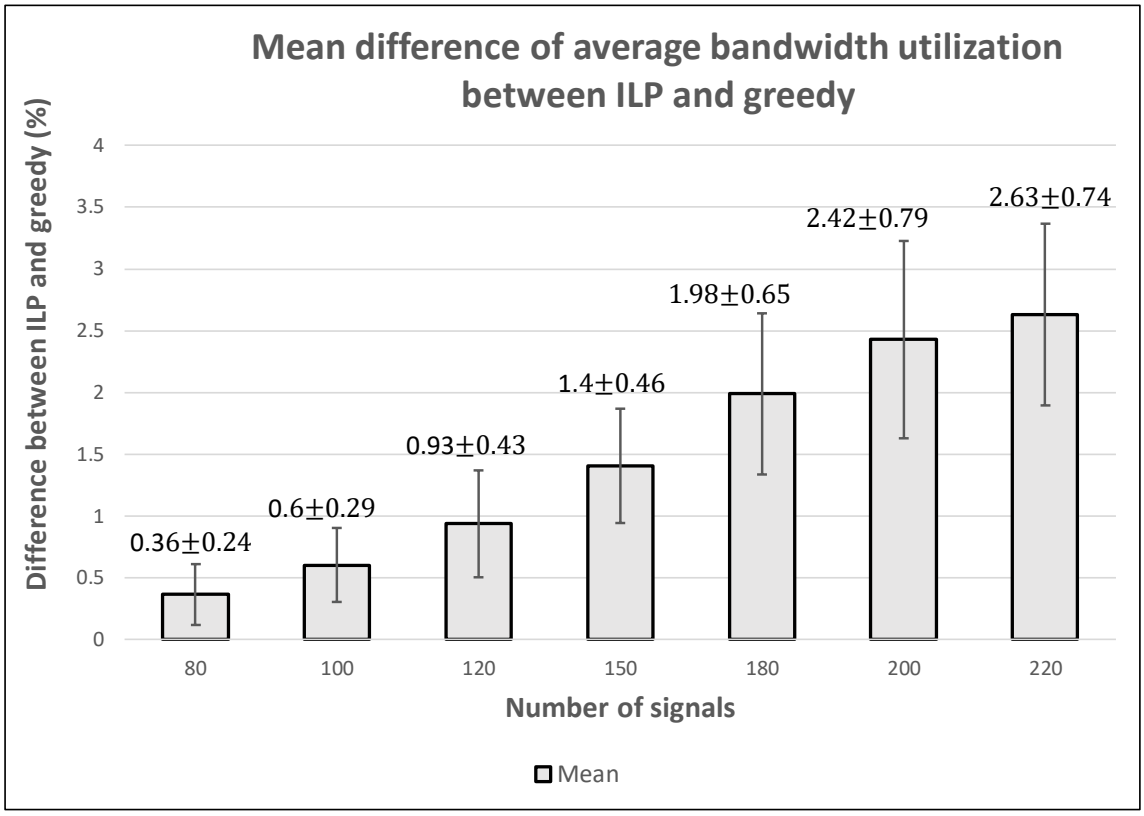


Figure 2.9: Average and standard deviation for differences on utilization between ILP and the greedy algorithm.

that with increasing number of signals, the difference in the bandwidth utilization given by the ILP and greedy approaches increases.

Figure 2.11 presents the average runtime of the systems for the ILP and greedy heuristic (in log scale). It is evident that the ILP would have scalability issues for larger systems, as the average runtime for the systems with 220 signals is already over 2.5 hours. The greedy approach on the other hand runs about 6 orders of magnitude faster than the ILP. Thus, we can conclude that the greedy algorithm provides a packing whose bandwidth utilization is comparable to that of the ILP with a much smaller runtime. We note that in Figure 2.11

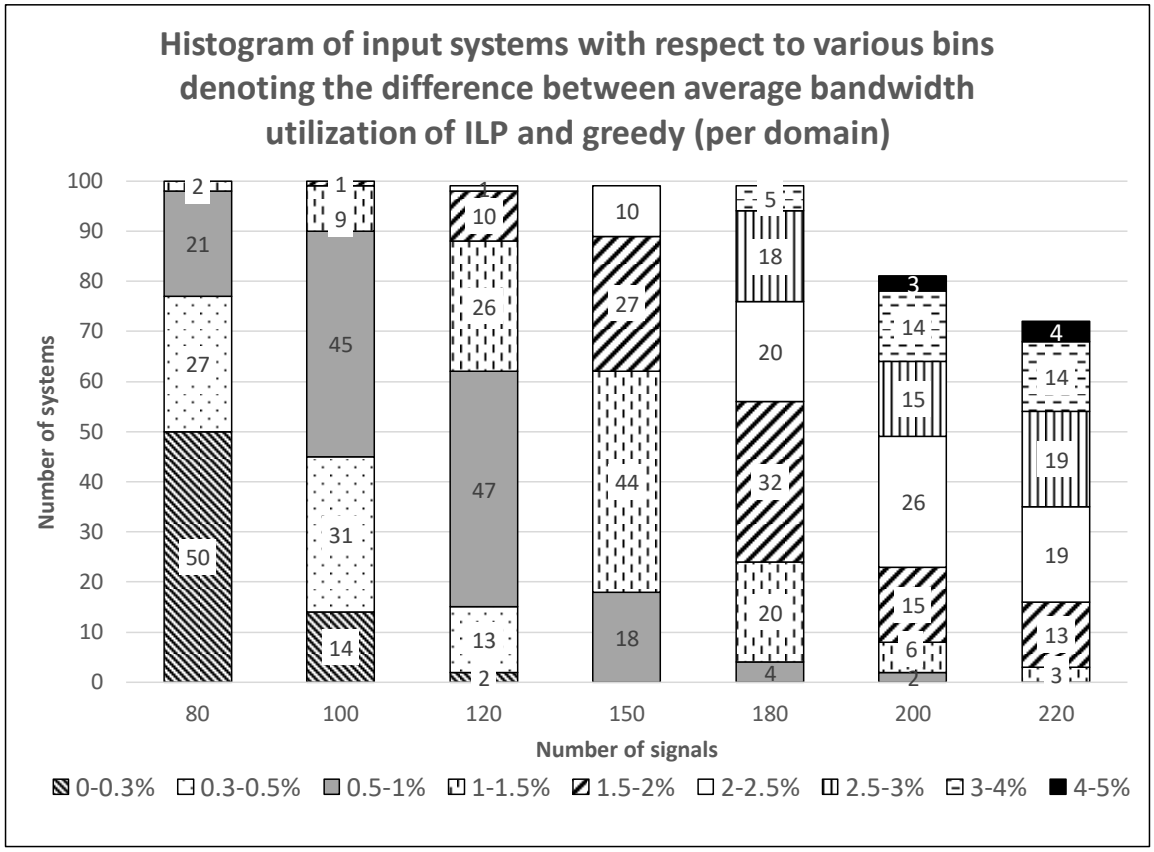


Figure 2.10: Distribution of systems within the various bin sizes (difference between greedy and ILP utilizations per domain).

the average runtime of the greedy heuristic for 100 signals is slightly higher than that for the subsequent case of 120 signals. This is because one of the systems (out of 100) turned out to be unschedulable and thus the heuristic runs 10 iterations for this case, thereby increasing the average runtime. On the other hand, the unschedulable system in the case of 120 signals becomes schedulable in just one iteration with our heuristic (please refer to Figure 2.13).

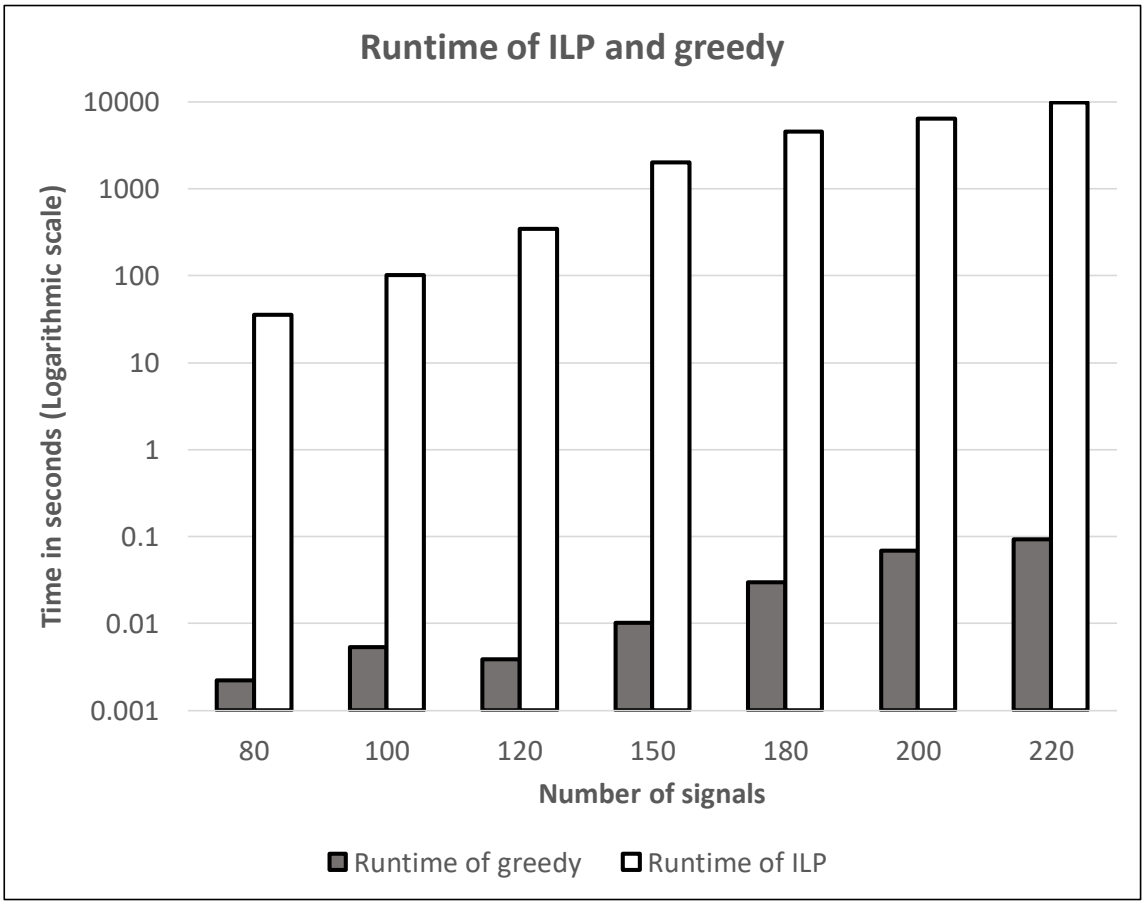


Figure 2.11: Greedy vs. ILP: runtime

Handling Infeasibility

In this set of experiments, we compare the approaches for handling infeasibility for the greedy heuristic. We generated 5000 random systems for each system size (number of signals in the system). As described in Section 2.6.2, we have implemented three variations of the repacking heuristic, namely "All", "Unassigned" and "Irreducible subset", and each of these variations uses three types of repacking algorithms: first-fit (FF), best-fit (BF) and worst-fit (WF). We also combined all the three variations (with first-fit), which resulted in (slightly) better

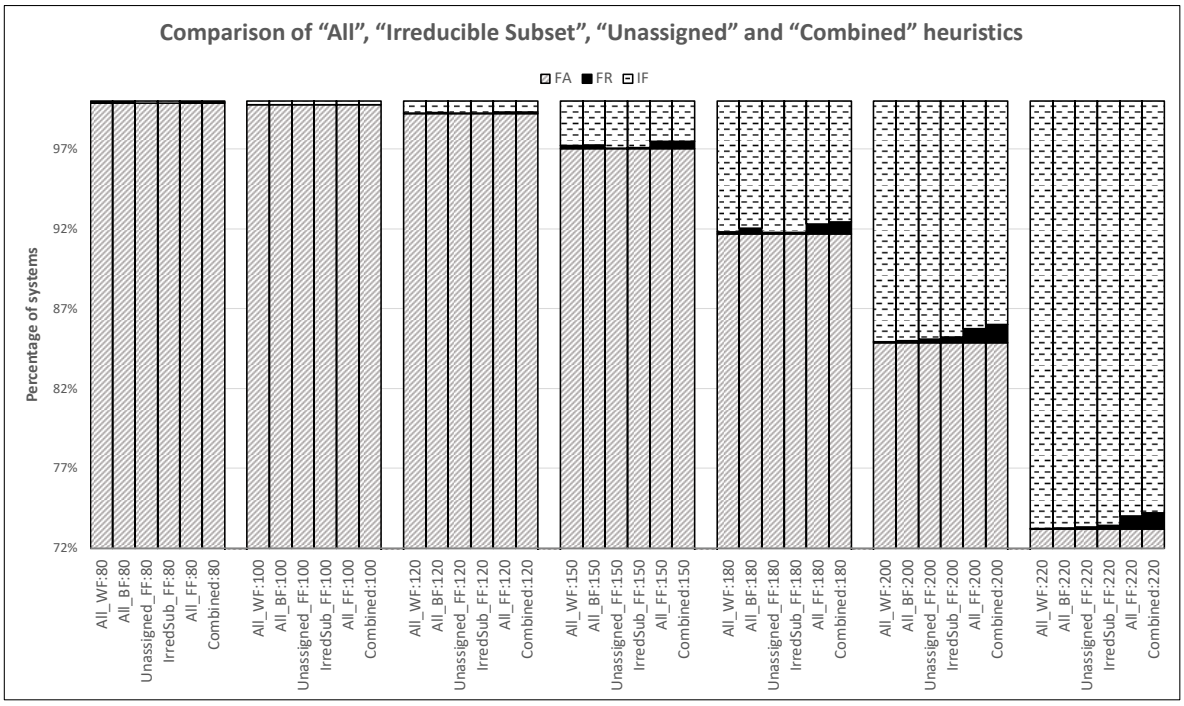


Figure 2.12: Comparison of “All”,“Irreducible subset”,“Unassigned” and “Combined” heuristics with respect to schedulability

results with respect to feasibility. We refer to this heuristic as the “Combined” heuristic.

In the experiments, we find that FF consistently outperforms the other two repacking algorithms BF and WF. Among the three variations, “All” heuristic gives the best results most of the time with respect to the number of feasible systems after the iterative procedure. To minimize clutter, in Figure 2.12 we only present the results for six approaches: “All” with FF, WF and BF, “Unassigned” with FF, “Irreducible Subset” with FF, and “Combined” with FF. The rectangular bars represent the percentage of the total number of systems, the gray section gives the number of systems feasible in first attempt (FA), the black section gives the number of systems feasible after repacking (FR), and the section with horizontal dash pattern gives the number of infeasible systems (IF).

We observe that the scheme “All” with first-fit (labeled as All-FF) is able to get the maximum

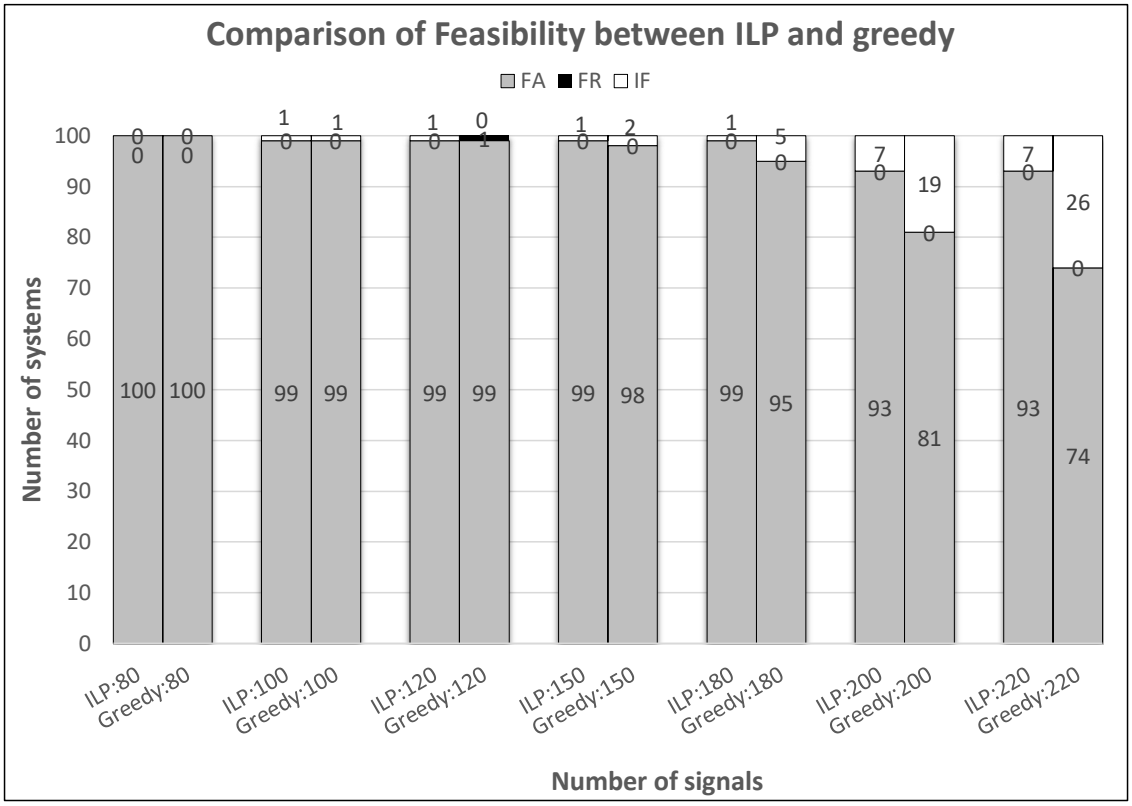


Figure 2.13: Infeasibility handling of ILP and greedy heuristic

number of systems to become feasible after the iterative step (for each system size). This is due to the fact that the "Irreducible subset" and "Unassigned" heuristics unpack a subset of the frames unpacked by the "All" heuristic. Therefore "All" is able to remove potentially "problematic" signals from all the frames and thus provide more schedulable cases. Also, by combining all our repacking heuristics, more than 92% of the systems become feasible (after repacking) for signal sizes 180 and below. For larger signal sizes, namely 200 and 220, 86% and 74% of the systems become feasible (after repacking) respectively.

Finally, we compare the infeasibility handling of the ILP and the greedy heuristic. For ILP, we use the iterative procedure described in Section 2.5.3, and for the greedy heuristic we

use the "All" with first-fit scheme, since it was found to give the best results for infeasibility handling. Due to the long runtime of ILP, we generated 100 random systems for each system size. As in Figure 2.13, the performance of the greedy heuristic is comparable to that of ILP with respect to the number of feasible cases for small systems (namely, with number of signals below 150). However, for system size of 180 the greedy packing results in about 5% infeasible cases whereas the ILP delivers just 1% infeasible cases. Furthermore, for system size 200 and 220, the ILP gives 7% infeasible cases whereas the greedy approach gives 19% and 26% infeasible cases respectively. Due to its optimal packing (lower bandwidth utilization over the network), the ILP provides better feasibility at the cost of a much longer runtime.

2.8.2 Multi Domain CAN-FD System with ACGM

The setup for our experiments with the ACGM is the same as the one that we used for the BCGM. We implemented our algorithms in C++. For all the experiments discussed in this section, we generated 5000 benchmarks (refer Section 2.8.1 for generation of synthetic systems) for each system size (in terms of the number of signals). In our experiments, methods Greedy-ACGM, Greedy-No-Cross-ACGM, Greedy-Decompose-Consideration-ACGM and Greedy-No-Constraint-ACGM (described in Section 2.7) for the multi-domain system with ACGM are compared against the multi-domain system with BCGM, denoted as "Greedy-BCGM". We use two performance parameters in the comparison: 1) Bandwidth Utilization per domain and 2) Number of schedulable systems. The runtimes for all the heuristics are quite comparable (please refer to Figure 2.6).

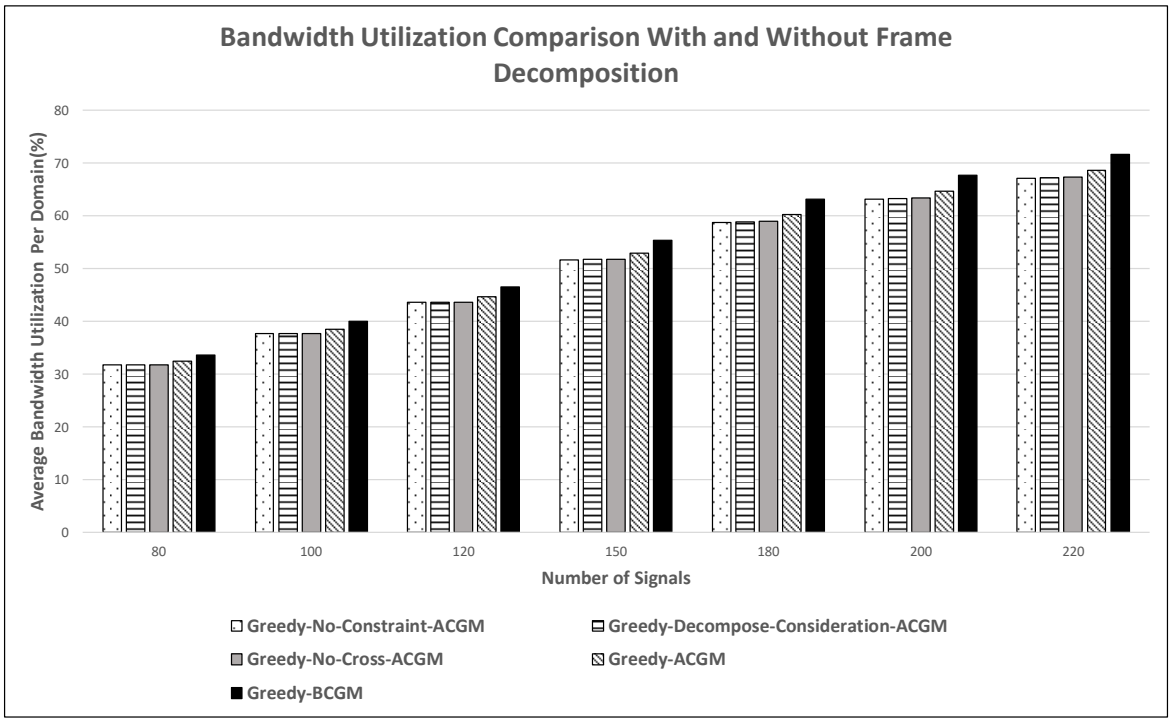


Figure 2.14: Comparison of the bandwidth utilization with and without Advanced Gateway.

Comparison of Bandwidth Utilization With and Without Decomposition at CGM

We first evaluate the different greedy decomposition heuristic approaches by comparing the bandwidth utilization they provide after packing and decomposing at the central gateway. Figure 2.14 presents the average bandwidth utilization per domain (as a percentage) for each signal size. The bandwidth utilization is averaged for those systems which are schedulable in the first attempt (without repacking). As before, the number of signals takes on the following values: $80, 100, 120, 150, 180, 200$ and 220. For this experiment, we consider 3 domains and 10 processors in each system.

As seen from the figure, there is an evident reduction in bandwidth utilization from the case of having a BCGM (i.e., no decomposition at the central gateway) to the other cases

where we assume the presence of an ACGM (i.e., with decomposition at the central gateway). Heuristics Greedy-No-Cross-ACGM, Greedy-Decompose-Consideration-ACGM and Greedy-No-Constraint-ACGM have almost similar performance in terms of bandwidth utilization. They perform better than Greedy-ACGM and Greedy-BCGM due to the following reasons:

1. Greedy-No-Cross-ACGM packs signals without considering the cross-domain aspect and Greedy-No-Constraint-ACGM packs signals without the harmonicity constraint for the signals in a frame. Thus, both of these methods achieve a more compact packing over the source domain. Further, due to the decomposition step, no replication occurs over the intended destination domains. Therefore, these methods help in reducing total bandwidth utilization.

2. Greedy-Decompose-Consideration-ACGM naturally performs better than Greedy-ACGM as the former packs signals into frames with the full knowledge of how a frame gets decomposed at the central gateway.

We observe an improvement in the average bandwidth utilization of about 4.5% per domain between Greedy-BCGM and Greedy-No-Constraint-ACGM, in the case with 220 signals. Thus, the total improvement over the three domains is about 13.5%, which is a significant reduction.

In order to further evaluate the proposed decomposition scheme, we also ran our decomposition heuristics on systems with larger number of domains, processors and signals. We generated synthetic systems having 500, 800 and 1000 signals with 5, 6 and 8 domains and 5, 7 and 10 ECUs per domain respectively. (As before, 5000 systems were used for each signal set size). For these experiments we used a slightly modified data generation scheme, where the contribution of signals from periods $1, 2$ and 5 ms which was 10% was distributed to signals with periods 10 and 20 ms equally. This modification was required as the original

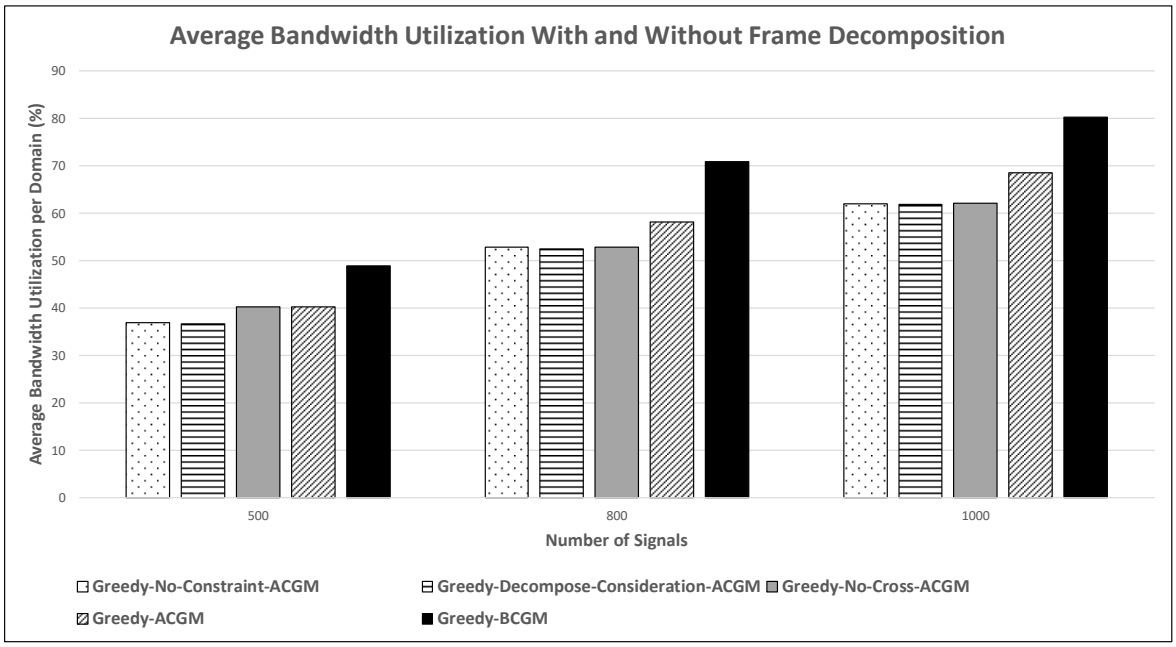


Figure 2.15: Comparison of the bandwidth utilization with and without ACGM (Large Systems).

data generation (described in Section 2.8.1) was close to saturating the buses with just 220 signals (as observed in Figure 2.5 the bandwidth utilization is about 72%.).

We plot the average bandwidth utilization per domain (for these large systems) in Figure 2.15. We observe that the decomposition schemes reduced the bandwidth utilization by about 18% compared to the Greedy-BCGM scheme. This is indeed a significant improvement in the bandwidth utilization. The runtime of these systems was observed to be less than 6 seconds per system (even in the case of the largest signal sets with 1000 signals), which shows that the heuristics do scale to systems with a large number of signals.

Comparison of Schedulability With and Without Frame Decomposition at CGM

In this experiment, we compare the performance of the decomposition heuristics (Greedy-ACGM, Greedy-No-Cross-ACGM, Greedy-Decompose-Consideration-ACGM and Greedy-No-Constraint-ACGM) with the 'Greedy-BCGM method on the basis of the number of schedulable systems. We run the schedulability analysis for the same 5000 synthetic systems

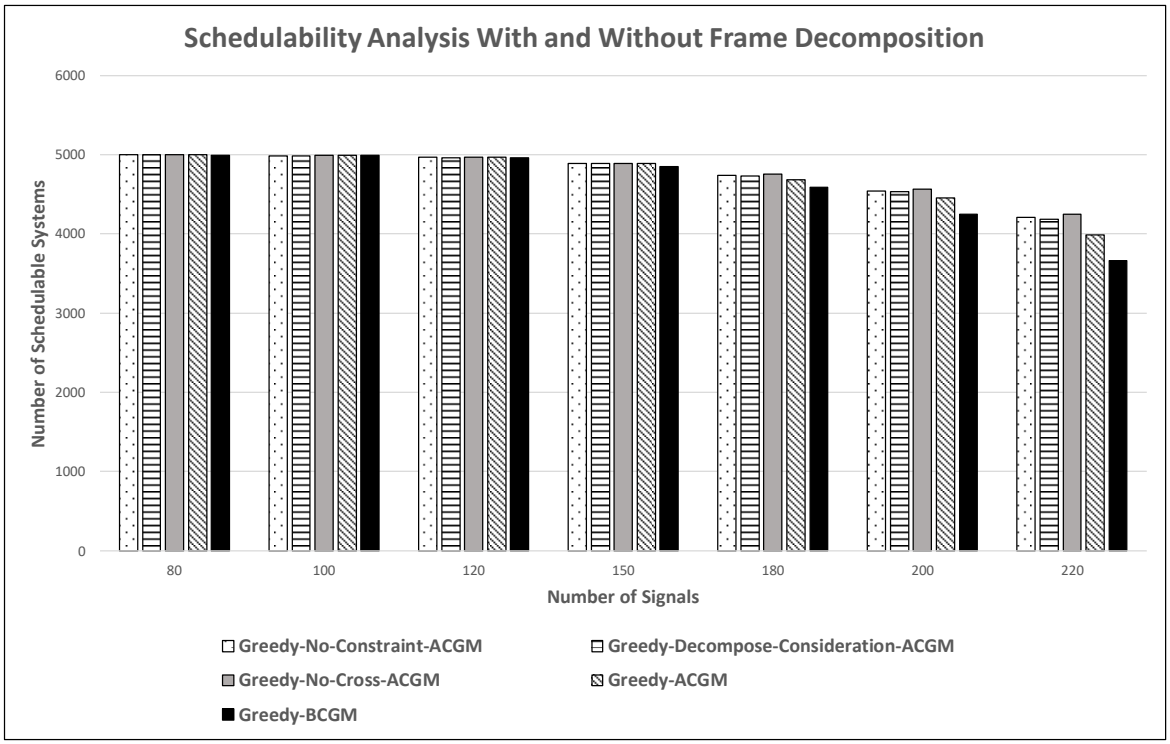


Figure 2.16: Comparison of the Schedulability with and without ACGM.

each with 3 domains and 10 processors, used above. In Figure 2.16 we plot the average number of systems (out of 5000) that are schedulable in the first attempt (without the repacking step) for each of the heuristics Greedy-ACGM, Greedy-No-Cross-ACGM, Greedy-Decompose-Consideration-ACGM and Greedy-No-Constraint-ACGM; we also compare them against the Greedy-BCGM heuristic. Since the repacking step is the same as before (refer

Section 2.8.1) in this experiment we do not report the number of systems corresponding to "FR" (Feasible after Repacking) and "IF" (Infeasible) here.

We observe that the heuristics for ACGM also perform better in terms of schedulability for systems with 150 or more signals. This is due to the fact that as the overall bandwidth utilization reduces (due to the non-replication of some signals over multiple domains), the available frames have lower response times (due to smaller payloads); this enables the improvement in schedulability. We observe that due to the decomposition at the ACGM, we are able to obtain about 11% more schedulable systems in the case of 220 signals. Also, Greedy-No-Cross-ACGM, Greedy-Decompose-Consideration-ACGM and Greedy-No-Constraint-ACGM are comparable in terms of schedulability; they perform better than Greedy-ACGM and Greedy-BCGM.

2.9 Summary

In this chapter, we motivate and propose solutions for the frame packing problem for multi-domain CAN-FD systems connected by 'Basic' and 'Advanced' gateways. Existing work on frame packing for CAN-FD systems has not considered the problem from multi-domain or advanced gateway (which may decompose frames) perspectives. Our experiments show the significance of considering the multi-domain aspect (i.e., the inter-domain communication) for packing signals into frames. We also present experimental results to show that the advanced gateway can further improve bandwidth utilization. We consider the problem for two types of gateways, and present solutions for both cases. For the 'basic' gateway, we propose an ILP-based approach and a greedy heuristic, both of which pack signals into frames with the goal of minimizing the bandwidth utilization over all the domains. In addition, we propose an extension to Audsley's algorithm for assigning priority identifiers to the frames

in the multi-domain case. In case of infeasibility, we develop several repacking heuristics so that the system may become feasible. Our experimental results show that the performance of the greedy heuristic is comparable to that of the ILP with respect to the bandwidth utilization as well as feasibility of the packed frames. However it is much faster than the ILP. For the 'advanced' gateway, we propose four heuristics which perform a signal-to-frame packing along with a decomposition at the central gateway. Our experimental results show that many of these heuristics improve both bandwidth utilization and schedulability.

Chapter 3

Offset Assignment to Signals for Improving Frame Packing in CAN-FD

3.1 Introduction

The signal-to-frame packing (or in short, *frame packing*) problem in CAN and CAN-FD has been studied in the literature and shown to be a challenging problem in Chapter 2 (and in the literature, [5, 47, 59, 71]). Efficient frame packing enables better bus utilization thereby increasing the amount of data that can be transmitted. In return, this provides for better system extensibility (i.e., the ability to accommodate future functionalities). However, all the existing works assume no signal offset assignment even when signals of different periods are packed into the same frame.

The current practice of frame packing is that each CAN frame has a fixed payload size (even if some of the signals may not be transmitted each time the frame is activated). This is supported by the automotive OSEK/AUTOSAR [23] communication layer specification and by commercial tools such as CANdb++ [74]. It eases the unpacking of frames at the receivers since the signals have a fixed location within the frame. This practice, combined with the existing work that has no consideration of signal offset assignment, means that the frame payload has to be no smaller than the sum of payloads from all the signals. In this work we propose to carefully distribute the signals into the frame instances to reduce the frame

payload and consequently the bus bandwidth. We term this problem as *offset assignment* to signals where an *offset* denotes a displacement (in terms of time) of the signal from the first instance of the frame. We provide a motivational example below.

3.1.1 Motivational Example

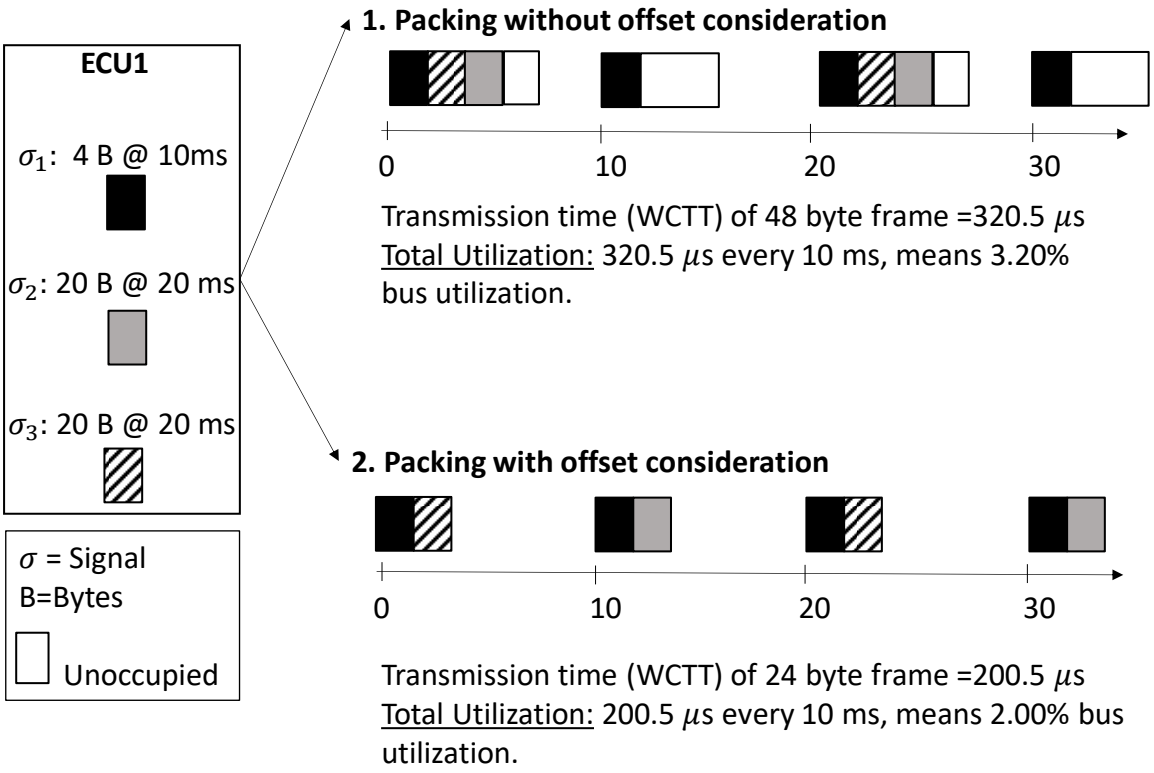


Figure 3.1: An example to motivate signal offset assignment for improving bandwidth utilization in frame packing.

Consider a CAN-FD system with a single bus and one ECU transmitting three signals as shown in Figure 3.1. The signal payload sizes and periods are also shown in the figure.

We compare two packing solutions for the signals. In the first solution, signal offsets are not considered; i.e., all the signals have an implicit offset of 0 ms. The resulting frame payload size is 48 bytes since the signal payload sizes $(4, 20, 20$ bytes) sum up to 44 bytes

and the nearest available CAN-FD payload is 48 bytes. The frame is transmitted every 10 ms, thus incurring a bandwidth utilization of 3.2% (as explained in Figure 3.1). Due to the discontinuity of CAN-FD size, there is a payload wastage (as denoted by the white box in the figure) of 4 bytes for frame instances at $20k$ ms (when all the signals are transmitted), and of 44 bytes for instances at $(20k + 10)$ ms (when only σ_1 is transmitted), where k is any non-negative integer.

In the second solution, signal σ_2 with a period of 20 ms is assigned an offset of 10 ms. The remaining two signals (σ_1 and σ_3) are assigned an offset of 0 ms. Thus, the two signals with period 20 ms are transmitted alternately with the signal σ_1 in the frame. The frame payload size is reduced to 24 bytes. This arrangement leads to zero payload wastage for the frame and a lower bandwidth utilization (as compared to the first case) of 2%. Thus, by assigning appropriate offsets to signals, we can improve the bandwidth utilization of a frame.

The above example shows that signal offset assignment can lead to an improvement in bandwidth utilization even with just one ECU and three signals. We deliberately choose a simple example to quickly convey the usefulness of offset assignment. In real systems, there are multiple CAN-FD buses and many ECUs, with each ECU generating signals that result in many frames. Hence, offset assignment can lead to a considerable improvement in bandwidth utilization in practice. In our experiments, we observe an improvement in bandwidth utilization of about 10.54% on a single bus.

3.1.2 Related Work

In this work, we study the frame packing problem when suitable offsets may be assigned to signals in a frame. Prior work has not considered the use of signal offsets in the context of CAN-FD frame packing. The SOAP shares several characteristics with the makespan

minimization problem (MMP) (also known as the *load balancing problem*), which arises in parallel computing. However, the presence of signals with different periods adds to the complexity and hence we cannot directly use any of the existing heuristics for MMP from literature to solve SOAP. We discuss the existing heuristics and approximation schemes for the MMP here. In general, the goal of MMP is to distribute a set of independent jobs with known execution times on multiple processors so that the makespan (i.e., the maximum completion time on any processor) is minimized. Graham [20] shows that a simple greedy algorithm for MMP provides a solution within twice the optimal value. Further, in [21], he shows that when the jobs are sorted in decreasing order of execution times, the greedy approach gives a solution within a factor of 4/3 of the optimal value. Hochbaum and Shmoys [27] present a polynomial time approximation scheme for MMP which, for any $\epsilon > 0$, provides a solution within the factor $(1 + \epsilon)$ of the optimal value.

It should be noted that our proposed offset assignment to **signals** is a concept that is different from the offset assignment to **frames** in [22]. In the latter, the offset assignment to frames is similar to task offset assignment that is proposed in real-time scheduling theory [52] to improve system schedulability. Unlike our work, in [52], [22] offsets are assigned to software tasks and frames, i.e., the scheduling entity. The main benefit of signal offset assignment is improved bus bandwidth utilization, although we observe some improvement in the schedulability of frames as a by-product of the main goal. In the frame packing context, we ensure the real-time schedulability of the packed frames using the analysis given by Davis et al. in [10], a correction to the original analysis [69].

3.2 Offset Assignment Problem for CAN-FD Frame Packing

In this section, we formulate the problem of *offset assignment* for signals in a frame. As input, we are given a set of ECUs and a set of *periodic* signals from all the ECUs which are connected via a CAN-FD bus. The goal is to pack the given signals into frames and assign suitable offsets to signals such that the bus bandwidth utilization is minimized. Thus, we have a main problem of packing signals into frames (*frame packing*) such that bandwidth utilization is minimized and a sub-problem of *offset assignment* to signals within a frame to augment the main goal.

Frame Packing in CAN-FD: Let $\Psi = \{\psi_i : i = 1, 2, \ldots, |\Psi|\}$ denote the set of ECUs. The set of signals from an ECU ψ_i is given by $\mathbf{S}(\psi_i) = \{\sigma_j^i \ : \ j = 1, 2, \ldots, |\mathbf{S}(\psi_i)|\}$. Each signal $\sigma \in \mathbf{S}(\psi_i)$ is specified using a triplet of parameters $\langle t(\sigma), d(\sigma), p(\sigma)\rangle$ which denote the period (in ms), deadline (in ms) and payload size (in bytes) of the signal respectively.

The desired output is a set of frames where each frame consists of a subset of signals from one ECU. The output set of frames is denoted as $\Gamma = \{\gamma_1, \gamma_2, \ldots, \}$, and each frame γ is characterized by a tuple $\langle\, \mathbf{S}(\gamma), T(\gamma), D(\gamma), P(\gamma), C(\gamma), \pi(\gamma)\,\rangle$, where $\mathbf{S}(\gamma)$ is the set of signals packed into γ. The quantities $T(\gamma)$, $D(\gamma)$, $P(\gamma)$, $C(\gamma)$ and $\pi(\gamma)$ are respectively the period, deadline, payload size, WCTT, and priority of the frame γ. (Additional information about these parameters appears later in this section). All the frames must satisfy the following properties: a) Each signal σ is placed in exactly one frame γ, and b) For each frame γ, all the signals in γ are from the *same* ECU.

Signal Offset Assignment: A frame may consist of a group of signals with different periods. For every frame having signals with different periods, each signal $\sigma \in \mathbf{S}(\gamma)$ is

assigned an offset from its set of *permissible offsets*. As mentioned earlier, the goal of offset assignment is to minimize the bandwidth utilization. To specify the conditions for a valid offset assignment, we need to introduce a few definitions.

- The *base period* t_b for a set $\mathbf{S}(\gamma)$ of signals with two or more distinct periods is defined as the greatest common divisor (gcd) of the signal periods. That is, $t_b = \text{gcd}\{t(\sigma) : \sigma \in \mathbf{S}(\gamma)\}$.

- The *hyperperiod* t_h for a set $\mathbf{S}(\gamma)$ of signals is defined as the least common multiple (lcm) of all the periods; that is, $t_h = \text{lcm}\{t(\sigma) : \sigma \in \mathbf{S}(\gamma)\}$. Note that when the signal periods are harmonic, t_b is the smallest and t_h is the largest period in a signal set.

- Each frame γ has a tuple $F(\gamma) = \langle F_0, F_1, ..., F_{N(\gamma)-1} \rangle$ of *frame instances*, where $N(\gamma) = t_h/t_b$ denotes the total number of instances of γ in the hyperperiod t_h. The *activation time* $A(F_n)$ of frame instance $F_n \in F(\gamma)$, where $n \in \{0, 1, .., N(\gamma) - 1\}$, is given by $A(F_n) = n \times t_b$.

We are now ready to state the conditions to be satisfied by a valid offset assignment. Let t_1, t_2, ..., t_K denote the periods of signals in $\mathbf{S}(\gamma)$, where K is the number of distinct signal periods.

- For a signal σ with period $t_i = t(\sigma)$, the set of *permissible offsets* is given by $\Phi(t(\sigma)) = \{j \times t_b : j = 0, 1, .., N_i - 1\}$ where $N_i = t(\sigma)/t_b$. Thus, signal σ must be assigned an offset $\phi(\sigma) \in \Phi(t(\sigma))$. For values of $j > (N_i - 1)$, the offsets are repeated; hence, we do not consider them. Note that for a signal σ with period equal to the base period (i.e., $t(\sigma) = t_b$), the set of permissible offsets is $\Phi(t_b) = \{0\}$.

- Each signal σ is transmitted in $t_h/t(\sigma)$ frame instances for any valid offset assignment: σ is assigned to all frame instances F_n such that $A(F_n) = \phi(\sigma) + (q - 1) \times t(\sigma)$ for

$q \in \{1, ..., \frac{t_h}{t(\sigma)}\}$. We use the notation $\sigma \in F_n$ to indicate that signal σ is assigned to frame instance F_n. Thus, the frame instances act like *bins* for signals, and each offset assignment represents an assignment of signals to a different set of bins.

From the above formulation of offset assignment, we can observe that assigning non-zero offsets to signals allows us to carefully balance the loads of frame instances. Given $\mathbf{S}(\gamma)$ and the offset of each signal in $\mathbf{S}(\gamma)$, the other parameters of the frame γ are determined as follows.

- The period $T(\gamma)$ of frame γ is equal to the base period of the signals in γ.
- $D(\gamma) = \min\{d(\sigma) \ : \ \sigma \in \mathbf{S}(\gamma)\}$; i.e., the deadline of γ is the smallest deadline among the signals in γ. All instances of a frame have the same deadline.
- The *occupancy* of a frame instance F_n is defined as the sum of its constituent signal payloads, i.e., $\sum_{\sigma \in F_n} p(\sigma)$. The payload $P(F_n)$ of the frame instance F_n is the smallest CAN-FD payload size that is $\geq$ the occupancy of F_n; thus, $P(F_n) \geq \sum_{\sigma \in F_n} p(\sigma)$. CAN-FD standard [25] restricts $P(\gamma)$ to be one of the following values: 0 through 8, 12, 16, 20, 24, 32, 48 and 64 bytes. The payload of a frame is taken as the maximum payload of all its frame instances, $P(\gamma) = \max \{P(F_n) : n = 0, 1, \ldots, N(\gamma) - 1\}$.
- The WCTT $C(\gamma)$ of a frame γ is determined by Equation (1.2), with the variable p being replaced by $P(\gamma)$.
- $\pi(\gamma)$ represents the unique priority assigned to frame γ. (Priority assignment is discussed later in this paper.)

The bandwidth utilization $U(\gamma)$ of frame γ is defined as

$$U(\gamma) \ = \ \frac{C(\gamma)}{T(\gamma)} \tag{3.1}$$

3.3 The Complexity of Offset Assignment

In this section, we state the decision version of the offset assignment problem (**SOAP**) and establish its complexity. Informally, given a set $\mathbf{S}(\gamma)$ of signals in frame γ which satisfy the conditions mentioned in Section 3.2, an offset assignment must ensure the following properties: (i) the number of frame instances per hyperperiod is $N(\gamma) = t_h/t_b$; and (ii) the maximum occupancy of a frame instance is minimized.

Here we use the occupancy of a frame as a proxy for its bandwidth utilization. There are two reasons for choosing this objective. First, the bandwidth of a frame γ is a monotonic non-decreasing function of the occupancy of γ. Hence, minimizing the occupancy also reduces the bandwidth. Second, in CAN-FD systems, unlike the occupancy (which is the sum of the signal payloads), the frame bandwidth is a discontinuous and nonlinear function of the signal payloads. In general, it is difficult to handle such functions in complexity proofs and approximation analysis.

A formulation of the decision version of SOAP is as follows. We define an ***instance*** for the problem which specifies all the input parameters for the problem. The decision version of the problem is stated as a ***question*** with a yes/no answer.

Definition 3.1. (SOAP) An instance of the SOAP consists of a set $\mathbf{S}$ of signals and an integer B.

Question: *Is there an assignment of the signals in* $\mathbf{S}$ *into frame instances satisfying the above conditions so that the occupancy of each frame instance is at most* B*?*

Our next result establishes the complexity of SOAP using a reduction from the 3-Partition problem stated below.

Definition 3.2. (3-Partition) An instance of the 3-Partition problem consists of pos-

itive integers m, Q and a set $\mathbf{A} = \{a_1, a_2, \ldots a_{3m}\}$ of $3m$ positive integers such that $Q/4 < a_i < Q/2$ for $1 \leq i \leq 3m$ and $\sum_{i=1}^{3m} a_i = mQ$.

Question: *Can* $\mathbf{A}$ *be partitioned into* m *subsets* $\mathbf{A}_1$, $\mathbf{A}_2$, ..., $\mathbf{A}_m$ *such that the sum of the elements in each subset* $\mathbf{A}_j$ *is equal to* Q, *for* $1 \leq j \leq m$*?*

It is known that 3-Partition is strongly **NP**-complete [18]. When there is a solution to a given 3-Partition instance, the constraints on the size of each integer a_i $(1 \leq i \leq 3m)$ ensure that each subset $\mathbf{A}_j$ has exactly *three* integers [18].

Theorem 3.3. *SOAP is strongly* ***NP****-complete even if the number of distinct signal periods is 2.*

Proof: It is easy to see that SOAP is in **NP** since given an offset assignment one can readily verify that the size of the frame instance at each of the offset values is bounded by the given value B. To prove **NP**-hardness, we use a reduction from 3-Partition defined above. Given a 3-Partition instance, the reduction produces a SOAP instance with $3m+1$ signals. One signal σ_0 has period 1 and size 1. All of the remaining $3m$ signals, denoted by σ_1, σ_2, ..., σ_{3m}, have period m. Thus, the hyperperiod $= m$ and the number of frame instances is also m. For convenience, we let $\mathbf{S}' = \{\sigma_1, \sigma_2, \ldots, \sigma_{3m}\}$. In any assignment, there are m instances of signal σ_0 and only one instance of each of the other signals. The offset for each of the $3m$ signals in $\mathbf{S}'$ must be in $\{0, 1, \ldots, m-1\}$. Further, signal $\sigma_i \in \mathbf{S}'$ has size a_i, $1 \leq i \leq 3m$. The maximum capacity B of each frame is set to $Q+1$. This completes the construction which can be carried out in polynomial time.

Suppose the 3-Partition instance has a solution. Consider the following solution to the SOAP instance. Choose the offset 0 for signal σ_0; for each signal in set $\mathbf{A}_i$, choose the offset value $i-1$, $1 \leq i \leq m$. To see that this satisfies the maximum occupancy constraint, note that at offset i, signal σ_0 contributes 1 to the size of the frame and the signals corresponding

to the items in $\mathbf{A}_{i+1}$ contribute a total size of Q, $0 \leq i \leq m-1$. Thus, the occupancy of the frame instances at each offset value is exactly $Q+1$. In other words, there is a solution to the SOAP instance.

Now suppose there is a solution to the SOAP instance. To prove that the 3-Partition instance has a solution, we use the following claim.

Claim 1: For $0 \leq i \leq m-1$, the total size of the signals with offset i equals $Q+1$.

Proof of Claim 1: The total size of the signals from $\mathbf{S}'$ is mQ. At each of the offsets (0 through $m-1$), signal σ_0 uses 1 byte. Thus, at each offset i, there is room for a subset of signals from $\mathbf{S}'$ with a total size of at most Q. Further, at any offset i, if the total size of the signals from $\mathbf{S}'$ is *less than* Q, then at some other offset j, the total size of the signals with offset j will *exceed* Q; in other words, the total size of the frame at offset j will exceed $Q+1$, contradicting our assumption that we have a solution to SOAP. The claim follows.

Given a solution to SOAP, a solution to 3-Partition is obtained by letting subset $\mathbf{A}_i$ contain exactly those items that correspond to signals from $\mathbf{S}'$ which were assigned offset $i-1$, $1 \leq i \leq m$. In view of Claim 1, the sum of the items in $\mathbf{A}_i$ is equal to Q, $1 \leq i \leq m$. Thus, we have a solution to 3-Partition, which completes our proof of Theorem 3.3. □

If all the signals have the same period, SOAP is trivial since by our assumption, all the signals are assigned the offset of zero. Thus, Theorem 3.3 shows that SOAP becomes computationally intractable as soon as we allow signals with even two periods. Also, the reduction from 3-Partition shows that SOAP is strongly **NP**-complete. Thus, there is no pseudo-polynomial time algorithm (i.e., an algorithm whose time complexity is a polynomial function of the size of the input and the maximum integer value in the input) for SOAP unless $\mathbf{P} = \mathbf{NP}$ [18].

3.4 A Generalized Approximation Framework (GAF) for SOAP

In this section, we present an approximation framework for SOAP. Before describing this framework, we recall a necessary definition. An approximation algorithm $\mathcal{B}$ for a minimization problem provides a (worst-case) **performance guarantee** of $\rho \geq 1$ if for every instance of the problem, the solution value produced by $\mathcal{B}$ is at most $\rho \times$ OPT, where OPT is the optimal solution value. The smaller the value of ρ, the better is the quality of approximation. For many optimization problems, approximation algorithms with good performance guarantees are known (see for example, [73]).

Our approximation framework for SOAP relies on approximation algorithms for the well known **makespan minimization problem** (MMP) to be defined below. Given an approximation algorithm with a performance guarantee of ρ for MMP, we prove that our framework provides a performance guarantee of at most ρK, where K is the number of different periods of input signals. After proving this result (Theorem 3.6), we point out how one can derive several approximation algorithms for SOAP using known approximation algorithms for MMP. As mentioned earlier, the goal of SOAP is to minimize the maximum occupancy of any frame. Our performance guarantee results are with respect to this objective.

We begin by defining the **Makespan Minimization Problem**[1] (MMP) for scheduling jobs on multiprocessor systems. In MMP, it is assumed that jobs are independent (i.e., there are no precedence constraints among the jobs) and that they are executed in a non-preemptive fashion. Given an assignment of jobs to processors, the **completion time** for a processor is the sum of the execution times of the jobs assigned to that processor. The **makespan** of the schedule is the *maximum* completion time over all the processors. A formal statement

[1]This problem is also known as the **Load Balancing Problem** in the literature (e.g., [35]).

of the MMP problem is as follows.

Definition 3.4. (Makespan Minimization Problem) (MMP) An instance for the MMP consists of a set $\mathbf{T} = \{T_1, T_2, \ldots\}$ of jobs where the processing time of job T_i is β_i (a positive integer), $1 \leq i \leq n$.

Required: *Find an assignment of each job in* $\mathbf{T}$ *to one of* m *(identical) processors so that the makespan is minimized.*

The decision version of MMP is known to be strongly **NP**-complete [18]. However, it has many approximation algorithms with good performance guarantees (e.g., [27, 35]).

To see the usefulness of MMP in obtaining an approximation algorithm for SOAP, consider a set of signals $\mathbf{S}_i$ such that all the signals in $\mathbf{S}_i$ have period t_i. With a slight abuse of notation, let $\Phi(t_i)$ denote the set of permissible offsets for signals with period t_i. We can think of each signal $\sigma \in \mathbf{S}_i$ as a job whose execution time is equal to $p(\sigma)$ (i.e., the payload size of σ). Further, each offset value in $\Phi(t_i)$ can be thought of as a processor. With this correspondence, it can be seen that assigning offsets to signals in $\mathbf{S}_i$ to minimize the maximum occupancy at any offset value corresponds to the MMP problem. While this observation is useful when all the signals have the same period, any heuristic for SOAP must consider signals with different periods. We now explain how our approximation framework exploits this connection between MMP and SOAP and handles signals of many periods.

Idea behind our approximation framework (GAF for SOAP): Given a frame γ with different period signals (an instance of SOAP) and an approximation algorithm $\mathcal{A}$ with a performance guarantee of ρ for MMP, our framework provides a performance guarantee of at most ρK, where K is the number of distinct periods of signals in the SOAP instance. Since we consider the application of SOAP for a particular frame, for the sake of simplicity we omit γ from the notations hereafter. Our framework uses Algorithm $\mathcal{A}$ for MMP as a black

box. To discuss the details, let $t_1 < t_2 < \ldots < t_K$ denote the K distinct periods of the input signals in increasing order. Let $\mathbf{S}_i$ denote the set of signals with period t_i, $1 \leq i \leq K$ in γ. The number of frame instances to be used is $N = t_h/t_b$ where $t_h = \text{lcm}\{t_i, i = 1, ..., K\}$ and $t_b = \text{gcd}\{t_i, i = 1, ..., K\}$. As mentioned earlier, let $\Phi(t_i)$ denote the set of *permissible offsets* for signals with period t_i. The set of all possible offset values is given by $\Phi = \{j \times t_b \ : \ 0 \leq j \leq N - 1\}$.

Our SOAP framework (in Figure 3.2) considers each set $\mathbf{S}_i$ of signals with period t_i ($1 \leq i \leq K$) *separately* and uses $\mathcal{A}$ to pack those signals as follows: Let $N_i = t_i/t_b$. As explained above, we treat each signal σ in $\mathbf{S}_i$ as a job with processing time $p(\sigma)$ (the payload size of σ) and the set $\Phi(t_i) = \{j \times t_b \ : \ 0 \leq j \leq N_i - 1\}$ of possible offset values for the signals in $\mathbf{S}_i$ as the set of processors to which the jobs must be assigned. Any solution to the resulting MMP represents a packing of the signals in $\mathbf{S}_i$. After a packing is returned by $\mathcal{A}$, to account for the periodicity of signals, we use the following step: for any signal $\sigma \in \mathbf{S}_i$ which is assigned offset $\phi(\sigma) \in \Phi(t_i)$, we place σ in the frame instances F_n such that $A(F_n) = \phi(\sigma) + (q - 1) \times t_i$, where $q = 1, 2, \ldots, t_h/t_i$ and $0 \leq n \leq N - 1$. When this process (Step 2(c) in Figure 3.2) is repeated for all the K period values, we get K separate offset assignments. In the last step, we merge all these assignments by considering each offset value $j \times t_b$ ($0 \leq j \leq N - 1$) separately and combining the signals from various periods which were assigned that offset value. Thus, in the final solution produced by our framework, for $0 \leq j \leq N - 1$, the occupancy of a frame instance with offset $j \times t_b$ is the sum of the occupancies of that offset value over all the periods.

To establish the performance guarantee for SOAP-Approx, we need to introduce some notation and prove a lemma. For the subset $\mathbf{S}_i$ of signals (having period t_i), let OPT_i denote the optimal solution value (i.e., the largest occupancy of any frame instance in any optimal solution) for SOAP restricted to subset $\mathbf{S}_i$, $1 \leq i \leq K$. Let OPT denote the optimal solution

Steps of SOAP-Approx:

1. Let $\mathbf{S}_i$ be the set of signals with period t_i $(1 \leq i \leq K)$.
2. **for** $i = 1$ **to** K **do**
   (a) Let $N_i = t_i/t_b$.
   (b) Use the approximation algorithm $\mathcal{A}$ for the signal set $\mathbf{S}_i$ with $\Phi(t_i) = \{j \times t_b : 0 \leq j \leq N_i - 1\}$ as the permissible offsets.
   (c) For each signal σ which is assigned offset $\phi(\sigma) \in \Phi(t_i)$ place σ in the frame instances F_n such that $A(F_n) = \phi(\sigma) + (q-1) \times t_i$, where $q = 1, 2, \ldots, t_h/t_i$ and $0 \leq n \leq N - 1$.
   (d) Let P_i denote the resulting assignment for $\mathbf{S}_i$.
3. Merge the K assignments P_1, P_2 ..., P_K into a single offset assignment P by combining all the signals with the same offset in different assignments into the same frame instance. Output the assignment P.

Figure 3.2: Algorithm SOAP-Approx

value for SOAP for the set $\mathbf{S}$. The following lemma shows a relationship between OPT and OPT_i for $1 \leq i \leq K$.

Lemma 3.5. *For* $1 \leq i \leq K$, $\text{OPT}_i \leq \text{OPT}$.

Proof: For any $i \geq 1$, we prove by contradiction that $\text{OPT}_i \leq \text{OPT}$. Suppose for some i, $\text{OPT}_i > \text{OPT}$. Consider any optimal solution for set $\mathbf{S}$ and remove from that solution all the signals except those from $\mathbf{S}_i$. The result is a solution for $\mathbf{S}_i$. Note that the removal procedure does not increase the occupancy of any frame. Therefore, we have a solution for $\mathbf{S}_i$ where the maximum occupancy is *less than* OPT_i. This contradicts the assumption that OPT_i is an optimal solution value for $\mathbf{S}_i$, and the lemma follows. □

We are now ready to establish the performance guarantee provided by SOAP-Approx.

Theorem 3.6. *Let* $\mathbf{S}$ *denote the given set of signals and let* K *denote the number of distinct*

periods of the signals in $\mathbf{S}$. *Let* ρ *denote the performance guarantee provided by* $\mathcal{A}$ *for MMP. Then, the following results hold: (1) When there are signals with base period,* SOAP-Approx *provides a performance guarantee of* $1 + \rho(K - 1)$. *(2) When there are no signals with base period,* SOAP-Approx *provides a performance guarantee of* ρK.

Proof: For the set $\mathbf{S}$, let OPT and APPROX denote the maximum occupancy of frames in an optimal solution and that produced by SOAP-Approx respectively. Similarly, for set $\mathbf{S}_i$, let APPROX_i denote the maximum occupancy of a frame produced by SOAP-Approx, $1 \leq i \leq K$. We divide our analysis into two cases: (1) when signals with base period are present and (2) when there are no signals with base period.

Case 1: Signals with base period are present. In this case, $t_1 = t_b$ and for the signal set $\mathbf{S}_1$ (consisting of all signals with period t_1), the offset assignment produced by an optimal solution and that by our algorithm are identical; that is, each signal in $\mathbf{S}_1$ appears at all the N possible offset values. Therefore,

$$\text{APPROX}_1 \;=\; \text{OPT}_1 \;\leq\; \text{OPT},$$

where the last step is a consequence of Lemma 3.5. For signal sets $\mathbf{S}_2$ through $\mathbf{S}_K$, Step 2(b) of SOAP-Approx (Figure 3.2) uses Algorithm $\mathcal{A}$ which provides a performance guarantee of ρ. Using this fact and Lemma 3.5, we have for $2 \leq i \leq K$,

$$\text{APPROX}_i \;\leq\; \rho \times \text{OPT}_i \;\leq\; \rho \times \text{OPT}.$$

Adding the inequalities for APPROX_i, $1 \leq i \leq K$, we get

$$\sum_{i=1}^{K} \text{APPROX}_i \;\leq\; \text{OPT} \times [1 + \rho(K - 1)]\,.$$

Because of the merging procedure used in the algorithm (Step 3 of Figure 3.2), we have

$$\text{APPROX} \;\leq\; \sum_{i=1}^{K} \text{APPROX}_i.$$

From the last two equations, we have

$$\text{APPROX} \;\leq\; \text{OPT} \times [1 + \rho(K-1)]\,.$$

Case 2: Signals with base period are absent. When there are no signals with period equal to the base period (i.e., $t_1 \neq t_b$), then the offset assignment for the set of signals $\mathbf{S}_1$ may not be optimal; however, the maximum occupancy is still within the factor ρ of the optimal value. Hence using the same argument as in Case 1, we have for $1 \leq i \leq K$,

$$\text{APPROX}_i \;\leq\; \rho \times \text{OPT}_i \;\leq\; \rho \times \text{OPT}.$$

Adding the inequalities for APPROX_i, $1 \leq i \leq K$, we get

$$\text{APPROX} \;\leq\; \sum_{i=1}^{K} \text{APPROX}_i \;\leq\; \text{OPT} \times \rho K.$$

This completes the proof of Theorem 3.6. □

In the automotive domain, the number K of distinct periods of the signals is typically a constant independent from the number of signals. For example, as indicated in [39], $K = 9$ in the real-world automotive benchmark. Moreover, the number of distinct periods of signals actually packed in a frame is even much smaller (due to the packing algorithm which minimizes bandwidth utilization). Hence, in practice, one would expect the approximation algorithm to perform better than what its worst-case performance guarantee would indicate.

3.4.1 Deriving approximation algorithms from the framework

Several approximation algorithms with proven performance guarantees are known for MMP. We now point out how our framework enables us to get different approximation algorithms for SOAP using known approximations for MMP.

(i) The simple greedy algorithm $\mathcal{A}_1$, which considers signals in an arbitrary order and assigns each signal an offset which corresponds to a frame instance that currently has the least load, provides a performance guarantee of 2 for MMP [21, 35]. Using this algorithm in Step 2(b) of Figure 3.2, our framework leads to an approximation algorithm for SOAP with a performance guarantee of $2K - 1$ when there are signals with base period and $2K$ otherwise.

(ii) The variant of the greedy algorithm (denoted by $\mathcal{A}_2$) which considers signals in *decreasing* order of sizes provides a performance guarantee of $4/3$ for MMP [21]. Using this algorithm, our framework leads to an approximation algorithm for SOAP with a performance guarantee of $(4K - 1)/3$ when there are signals with base period and $4K/3$ otherwise.

(iii) Hochbaum and Shmoys [27] present an approximation scheme (denoted by $\mathcal{A}_\epsilon$), which for any given $\epsilon > 0$ provides a performance guarantee of $(1 + \epsilon)$ for MMP. Using $\mathcal{A}_\epsilon$, our framework provides a performance guarantee of $[1 + (K - 1)(1 + \epsilon)]$ when there are signals of base period and $(1 + \epsilon)K$ otherwise. By choosing ϵ appropriately, this performance guarantee can be made arbitrarily close to K. Theoretically, this algorithm runs in polynomial time. However, as observed in [27], this is not a viable approach in practice for small values of ϵ since its running time is $O((n/\epsilon)^{1/\epsilon^2})$ (For example, for $\epsilon = 0.1$, the exponent of n in the running time is 100). Hence in our experiments we do not use $\mathcal{A}_\epsilon$.

(iv) One can formulate MMP as an integer linear program (ILP). This is shown in Section 3.4.2. The resulting ILP can be solved using any ILP solver to get an optimal solution to MMP. Let us denote this algorithm by $\mathcal{A}_3$. Using $\mathcal{A}_3$ (which provides a performance

guarantee of 1 for MMP), our framework provides a performance guarantee of K for SOAP. While the methods discussed in (i), (ii) and (iii) above run in polynomial time, the worst-case running time of the ILP-based method is not polynomial. However, there are many ILP solvers (such as [29]) that work very well in practice. So, this is indeed a viable practical approach.

3.4.2 An ILP Formulation for the MMP

In Section 3.3 we pointed correspondence between offset assignment and MMP when all the signals have the same period. This enables us to express this offset assignment problem as an ILP as follows.

Suppose we are given a set $\mathbf{S}$ of n signals denoted by σ_1, σ_2, ..., σ_n, all of which have the same period. Signal σ_i has a size of p_i (a positive real number), $1 \leq i \leq n$. We are also given a set of m frame instances denoted by F_1, F_2, ..., F_m. The goal is to assign each signal to a frame instance (which determines the offset) so that the maximum occupancy of among the frame instances is minimized.

The ILP formulation uses $\{0, 1\}$ valued variables x_{ij}, $1 \leq i \leq n$ and $1 \leq j \leq m$. The interpretation is that $x_{ij} = 1$ if signal σ_i is assigned to frame instance F_j; otherwise, $x_{ij} = 0$. We will use the real variable Δ to denote the value of makespan (which corresponds to the maximum occupancy of a frame). Now, the ILP formulation for MMP is as follows:

$$\text{Minimize} \quad \Delta$$

such that

$$\sum_{j=1}^{m} x_{ij} \;=\; 1, \quad \text{for all } i, \quad 1 \leq i \leq n \tag{3.2}$$

$$\sum_{i=1}^{n} x_{ij} \cdot p_i \;\leq\; \Delta, \quad \text{for all } j, \quad 1 \leq j \leq m \tag{3.3}$$

The set of constraints in Equation (3.2) ensures that each signal is assigned to exactly one frame instance. The second set of constraints (Equation (3.3)) ensures that the occupancy of each frame instance is at most Δ. Thus, an optimal solution to the ILP formulation corresponds to an optimal offset assignment for a set of signals with the same period.

3.5 Application of SOAP to Frame Packing

In order to show the efficacy of our proposed framework for SOAP, we apply it to a frame packing algorithm from literature. We choose the greedy algorithm discussed in [32], since the offset assignment step can be directly plugged into the bandwidth computation step (see Figure 3.3). In this algorithm, the frame payload is not fixed and it is computed on the basis of the frame packing. Using our offset assignment scheme, we try to minimize the frame payload by assigning suitable offsets to signals.

Our approach differs from the algorithm in [5] which constructs a dynamic programming table for each CAN-FD size to evaluate the best packing for every possible subset of signals. Hence, the CAN-FD payload is fixed for each evaluation; this makes the signal offset assignment ineffective since the main goal of offset assignment is to minimize the payload of a frame. Thus, the application of the offset assignment scheme to the algorithm in [5] requires more investigation and considerable modification. Hence, we propose to do that as part of our future work. Likewise, the approach by Di Natale et al. [47] is not suitable, as it formulates the frame packing problem in a single ILP formulation. It is not clear whether SOAP can be efficiently translated into ILP without knowing the frame packing. Besides, the applicability of [47] is already limited due to scalability issues.

Figure 3.3 presents an extension of the frame packing algorithm in [32] using our proposed SOAP approximation framework. As observed in [32], the frame packing part of the algorithm gives the best performance in terms of bandwidth utilization when signals are sorted in increasing order of signal period; hence, we use the same ordering in our implementation. However, we modify the algorithm to embed our offset assignment algorithm. In Figure 3.3 the frame packing approach (in [32]) is given on the left and our proposed SOAP framework is shown on the right. In each iteration of frame packing, the algorithm tries to add a signal σ to an existing frame or create a new frame (containing only σ). The bandwidth of each of these alternatives is evaluated after assigning the signal offsets (by calling the SOAP framework). If all the signals in a frame have same period, the bandwidth computation is straightforward since the offset for all the signals is 0. If the signals in a frame have two or more periods, we apply our proposed SOAP approach to improve the bandwidth utilization. For our experiments, we use algorithms $\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{A}_3$ (discussed in Section 3.4.1) for MMP to our SOAP framework and present the results in Section 3.8.

After obtaining a frame packing solution using offset assignment, we analyze the schedulability of the generated frames over the CAN-FD system. The schedulability analysis of CAN-FD system used by us follows that of CAN [10], which is discussed in Section 1.1.1.

The analysis is valid for our model of CAN-FD frames with offset assignment since the payload size of each instance of the frame is the same (namely, the maximum payload size of all the instances after offset assignment) and the deadline of each instance is the smallest among all the periods in a frame (as mentioned in Section 3.2). We use the well-known Audsley's algorithm [2] to assign priorities to all the frames such that each frame meets its deadline.

In case Audsley's algorithm reports unschedulability, we use a simple repacking heuristic which has the potential to make frames schedulable. Our repacking scheme follows that

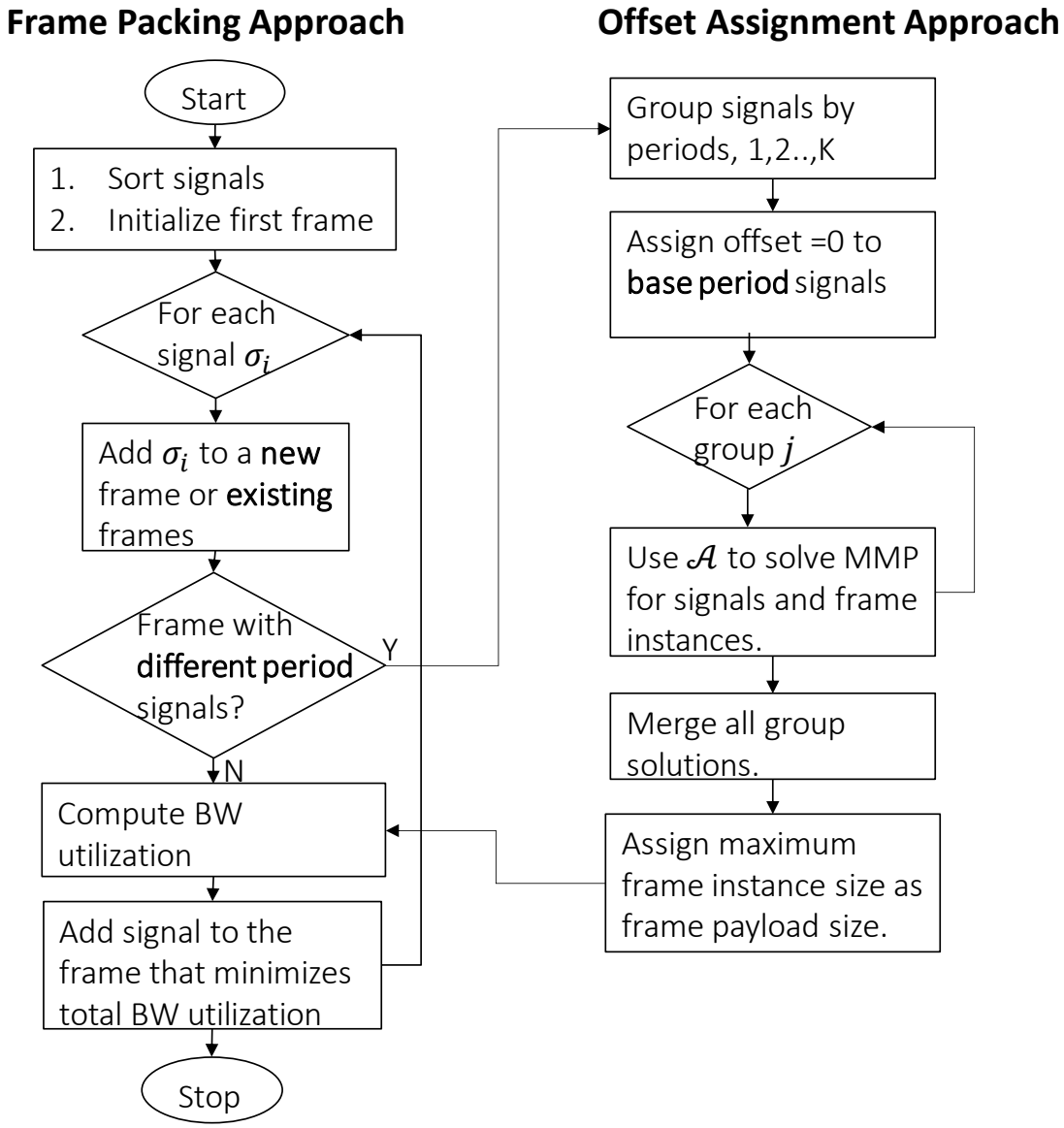


Figure 3.3: Flowchart describing the application of SOAP to a frame packing approach in CAN-FD

of [32]. When Audsley's algorithm reports unschedulability, we unpack the frames as follows: signals with the largest deadline in a mixed period frame are removed in order to reduce the frame payload size and hence its response time. The unpacked signals are then packed

using a *first-fit* scheme into an existing frame having deadline greater than or equal to the deadline of the signal; otherwise, they are packed into a new frame.

Practical Implementation of SOAP

In the case of no offset assignment to signals all the frame instances have the same constituent signals. However due to offset assignment to signals the different frame instances may have different signals. In order for the receiver to read the signals accurately the different frame instances need to be identified correctly at the receiving end.

In the conventional transmission scheme in AUTOSAR [1] for frames which have signals with offset assignment, the frame payload contains some additional bits to determine the specific frame instance. The number of additional number of bits depends on the number of unique frame instances. Figure 3.4 shows the conventional transmission of a frame with offset assignment. Since there are 4 different instances, they can be represented uniquely using 2 bits (added to the frame payload).

In order to improve the conventional transmission mechanism for offset assignment in AUTOSAR we propose the inclusion of 3 entities: 1) a "pre-ID" in the table (at the controller) that contains the information about the signal-to-frame packing, 2) a 'counter' at the receiver to keep a count of the frame instances and 3) a 'reset bit' in the frame payload. In the conventional scheme all the frame instances are mapped to the same I-PDU (Interface layer Protocol Data Units) [1] which are further assigned to the same frame. However in our proposed scheme we have different frame instances assigned to different I-PDUs. For example in Figure 3.5 there are 4 different I-PDUs that contain the 4 different signal sequences. Further, the different I-PDUs are mapped to a single frame. In the proposed scheme shown in Figure 3.5 the frame instances are transmitted sequentially and at each arrival time of the frame the counter (at the receiver) is incremented. Thus the counter determines the

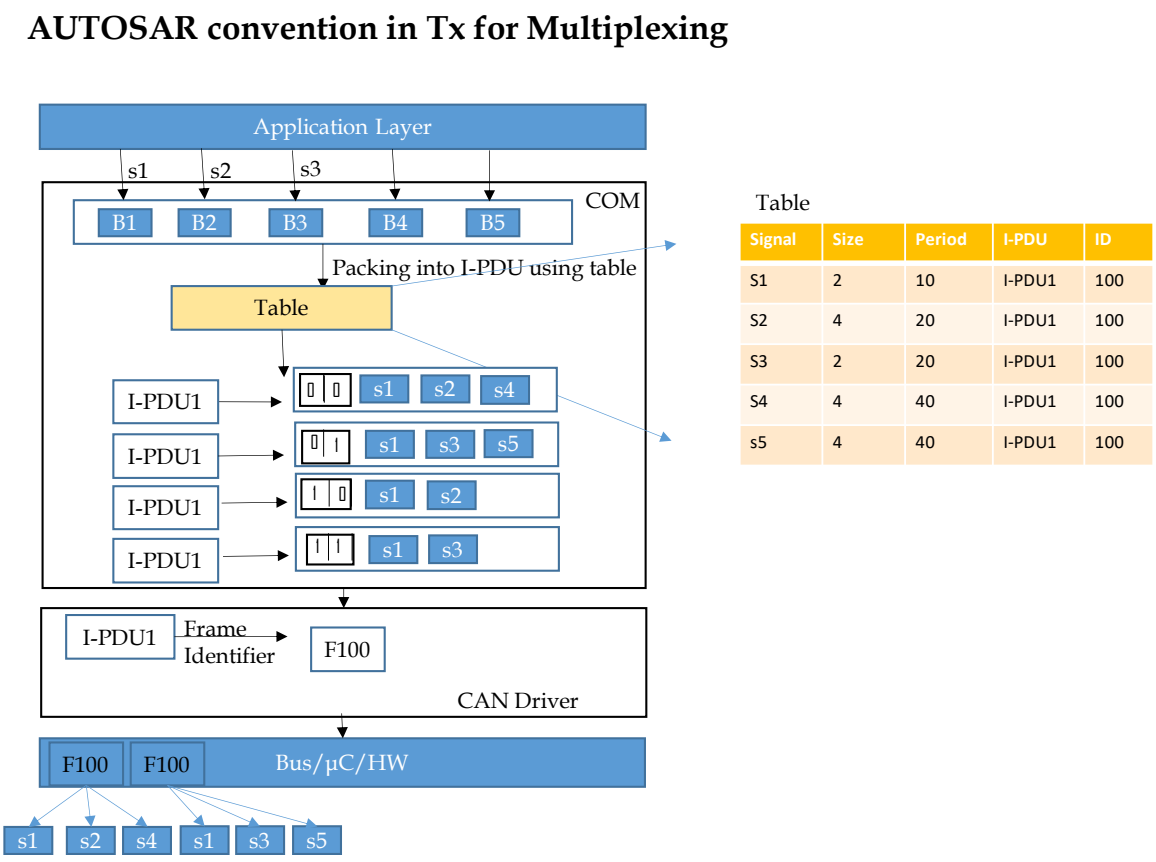
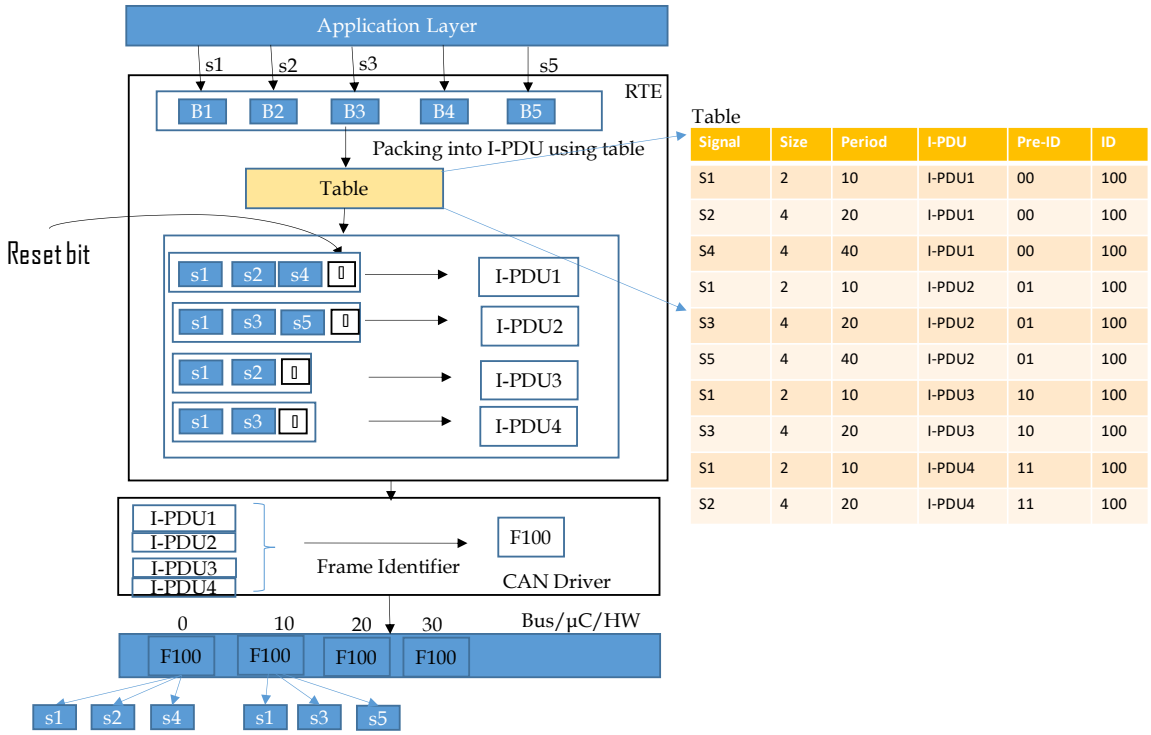


Figure 3.4: AUTOSAR Transmission Scheme for Signal Offset Assignment

appropriate I-PDU and in turn the constituent signals at the receiver. In case there is a transmission error at the sender (due to EMI), the reset bit is set and the frame is retransmitted starting from the first instance. Thus the reset bit is added in the frame payload so that the transmitter can convey about an erroneous transmission to the receiver. This mechanism does not load the bus bandwidth by including additional information about the frame instances into the frame payload (as done in the conventional scheme). Our scheme has an overhead of a single bit into the frame payload. This scheme has been filed for a patent by General Motors.

Proposed Tx for Multiplexing



Table

Signal	Size	Period	I-PDU	Pre-ID	ID
S1	2	10	I-PDU1	00	100
S2	4	20	I-PDU1	00	100
S4	4	40	I-PDU1	00	100
S1	2	10	I-PDU2	01	100
S3	4	20	I-PDU2	01	100
S5	4	40	I-PDU2	01	100
S1	2	10	I-PDU3	10	100
S3	4	20	I-PDU3	10	100
S1	2	10	I-PDU4	11	100
S2	4	20	I-PDU4	11	100

Figure 3.5: Proposed Transmission Scheme for Signal Offset Assignment

3.6 Improved Greedy Heuristics for SOAP

In addition to the above framework we propose two additional heuristics for SOAP.

1. **Greedy Offset Assignment Heuristic:** We propose a greedy offset assignment heuristic as shown in Figure 3.6. This heuristic is similar to GAF as it assigns offset to signals of each period from their permissible set of offsets. However unlike GAF the greedy heuristic does not create K different assignments which are then merged. Instead it starts with the complete set of instances of the frame, denoted as N. Then for each period i, where $1 \leq i \leq K$, the set of signals with period i are assigned offsets using an algorithm $\mathcal{A}$. After the assignment the corresponding signal payloads are

Greedy Offset Assignment Heuristic

(a) Let $\mathbf{S}_i$ be the set of signals with period t_i $(1 \leq i \leq K)$ and N be the total number of instances.

(b) **for** $i = 1$ **to** K **do**

i. Let $N_i = t_i/t_b$.

ii. Use the approximation algorithm $\mathcal{A}$ for the signal set $\mathbf{S}_i$ with $\Phi(t_i) = \{j \times t_b : 0 \leq j \leq N_i - 1\}$ as the permissible offsets.

iii. For each signal σ which is assigned offset $\phi(\sigma) \in \Phi(t_i)$ place σ in the frame instances F_n such that $A(F_n) = \phi(\sigma) + (q-1) \times t_i$, where $q = 1, 2, \ldots, t_h/t_i$ and $0 \leq n \leq N-1$.

(c) Report the final assignment.

Figure 3.6: Greedy Offset Assignment Heuristic

added to the rest of the frame instances according to the period i and the assigned offset. Thus in this heuristic the offset assignment for the different period groups is no longer independent of each other (as in GAF). This is precisely the reason why the greedy offset assignment heuristic performs slightly better in terms of bandwidth utilization than GAF.

2. **Interchange Heuristic:** The interchange heuristic is based on the multiprocessor scheduling algorithm described in [15]. However since the given algorithm was not directly applicable to SOAP due to the subtle differences between SOAP and MMP (as described in Section 3.4) we modified the existing algorithm to suit the requirements of SOAP. Figure 3.7 presents the algorithm of the Interchange heuristic. The signals are assigned offsets using any method to begin with. The payload difference between the frame instance with maximum occupancy and the frame instance with minimum occupancy, δ is computed. If there exists a signal in the maximum occupancy frame instance with size smaller than δ and if the minimum occupancy frame instance is

Interchange Heuristic

(a) Let N be the total number of instances of the frame. Assign offsets to each signal using any method.

(b) **while** interchange is possible **do**

i. Sort the frame instances in a decreasing order of payload.

ii. Compute $\delta = P(F_l) - P(F_k)$ where $P(F_l) = maxP(F_n),\ P(F_k) = minP(F_n),\ 0 \leq n \leq N-1$.

iii. **If** $\exists$ signal $\sigma_j \in F_l$ such that $p(\sigma_j) < \delta$ and $t_b * l \in \Phi(t_j)$ **then** *interchange* σ_j and its instances to $P(F_k)$. Go to Step(i).

iv. **Else** stop.

Figure 3.7: Interchange Offset Assignment Heuristic

a valid offset holder for the signal then the signal (along with all its instances) is *interchanged* between the two frame instances. This process is repeated until there can be no further interchange possible.

3.7 Improving SOAP by Using 2D Strip Packing Approach

With the focus on improving our approach for assigning offsets to signals in order to further optimize the frame packing in CAN-FD, we propose and develop a new framework that utilizes 2-dimensional strip packing approaches to solve for offset assignment. In the new framework we first propose a transformation function to convert the offset assignment (OA) instance to a 2D strip packing (SP) instance. Next we apply an existing SP approach and then transform the SP instance back to obtain an OA instance. We obtain a better approximation ratio (as compared to Section 3.4) for the new framework. We also show

the experimental results of using the new framework and compare them with the previous approaches. We observe an improvement in bandwidth utilization of about 1.69% as compared to our original framework, GAF (refer Section 3.4). We also apply the new framework over an automotive case-study. Although we observe modest improvements in bandwidth utilization, with the improved approximation ratio, the new framework may have potential for specific use cases. Hence we aim to continue this as future work.

2-Dimensional Strip Packing Problem

In a 2-D SP problem we are given a set of items, I. An item $m_i \in I$ has width w_i and height g_i. We have a strip of width W and infinite height. The width of the items is at most W. The objective of the problem is to pack all the given items in the strip such that the height used is minimized. The items cannot be rotated and no overlap of items is permitted.

To the best of our knowledge there is no work on offset assignment to signals for frame packing in CAN-FD which uses 2D SP. Our work on SOAP is the first one to talk about SOAP and its application to frame packing in CAN-FD [31]. In [44] the authors optimize the FlexRay static segment using a 2D bin packing approach. However in their problem they consider a fixed width and height and their objective is to minimize the number of bins packed, instead of the height used. Hence we cannot directly use their approach.

For 2D SP there are several algorithms that exist in literature. [43], [50] present a survey on 2-dimensional packing algorithms. In this work we use a *Bottom Left* algorithm [4], [7] for solving the SP problem. We also use an ILP formulation by [40] for strip packing, however we modify is suitably for offset assignment problem.

3.7.1 Problem Transformation

This section describes a one-to-one transformation of an offset assignment instance (refer 3.2) to the strip packing instance and vice versa. However the transformation function is valid when the signals have periods that are in a geometric series with the common ratio being a positive integer, $n > 1$. This is the main constraint for applying the SP approach to solve for OAP. In practice, this may not always be the case and in Section 3.7.3 we show how we tackle this constraint.

The offset assignment problem formulation is the same as described in Section 3.2 and hence we use the same notation. Let t_1, t_2, ..., t_K denote the periods of signals in $\mathbf{S}(\gamma)$, where K is the number of distinct signal periods. For describing the transformation function we assume that the periods of the signals are in a geometric series and their common ratio of is a positive integer n. Then for the **forward transformation** ($OA \rightarrow SP$) we convert each signal in the OAP to an item in SPP:

1. Each signal corresponds to an item, therefore, since $\mathbf{S}(\gamma)$ is a set of signals in frame γ, let $\mathbf{I}(\gamma)$ denote the set of items in frame γ.

2. The width w_i of item $m_i \in \mathbf{I}(\gamma)$ is given by the total number of instances of the corresponding signal $\sigma_i \in \mathbf{S}(\gamma)$ in the hyperperiod. Thus $w_i = t_h/t(\sigma_i)$.

3. The height g_i of item $m_i \in \mathbf{I}(\gamma)$ is the payload $p(\sigma_i)$ of signal $\sigma_i \in \mathbf{S}(\gamma)$.

4. The maximum width W of SP instance is t_h/t_b.

The **backward transformation** ($SP \rightarrow OA$) requires a transformation function T. We generalize and modify the transformation function defined in [44], which holds for FlexRay static segment (where the communication cycles are powers of 2).

We define the backward transformation function as follows:

$$T(x, y) = 0 \ \ if \ \ x = 0 \tag{3.4}$$

$$T(x, y) = T(\frac{x - r}{n}, \frac{y}{n}) + r * \frac{y}{n}, \ \ if \ x = r \tag{3.5}$$

Here $r = x\%n$. Thus $r = 0.., n-1$. In order to transform an SP instance to an OA instance, we map each slot on the X-axis in the SP instance to a unique slot in the OA instance (which corresponds to the activation time of the frame instance in OAP). The second largest width in the SP instance is taken as y, thus, $y = W/n$. The expression for conversion of each slot on the X-axis to an activation time for a frame instance is given by:

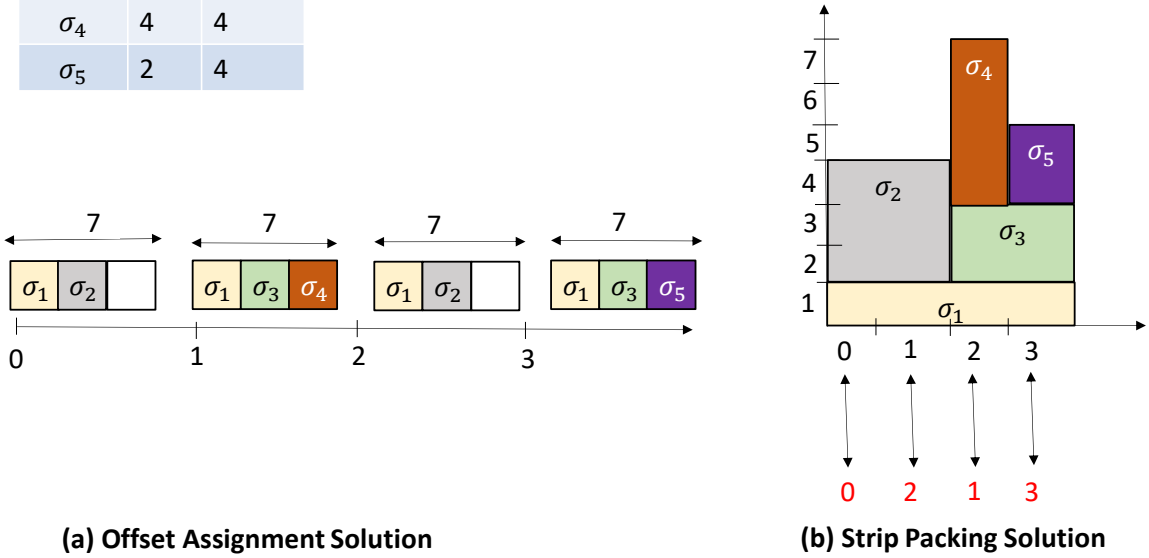
$$A(F_i) = T(x_k, W/n) * n \tag{3.6}$$

Here, the activation time A_i corresponds to the activation of frame instance F_i. If there is only one item in the SP instance, then $W = 1$, hence x has only one value, 0. Thus for every slot number $x_k, 0 \leq k \leq W - 1$ on the X-coordinate in SP, a unique activation time corresponding to OA is obtained. Since each $x \in \{0, 1, ...W - 1\}$ corresponds to a height, therefore by using the transformation, the payload of each frame instance is known. The maximum payload is taken as the frame payload (as described in Section 3.2).

In order to illustrate the transformation function consider Figure 3.8. The list of signals and their parameters are given in the table on the top left corner. For this example the value of $n = 2$ and $W = 4$. Figure 3.8(a) shows the offset assignment for the given signals. Signals σ_2 and σ_3 have a period of 2ms and they are assigned offsets 0 and 1 respectively. Thus σ_2 is added in frame instances 0 and 2, σ_3 is added to frame instances 1 and 3. Signals σ_4 and σ_5 have period of 4ms and they are assigned offsets of 1 and 3 respectively. σ_4 is in

frame instance 1 and σ_5 is present in frame instance 3. σ_1 is present in all the instances. The maximum size of the frame instance is 7 (instance at 0). The forward transformation

Signals	Size	Period
σ_1	1	1
σ_2	3	2
σ_3	2	2
σ_4	4	4
σ_5	2	4



(a) Offset Assignment Solution **(b) Strip Packing Solution**

Figure 3.8: Example to illustrate transformation function

$(OA \rightarrow SP)$ is straightforward; for example, the item corresponding to σ_1 has a width of 4 and height 1 units. The colors used for the signals are also used to denote the corresponding items.

In this work we use an existing approximation algorithm called Bottom Left (BL) [4] for 2-D strip packing. The BL algorithm first sorts the items in decreasing order of width and places the items on the bottom most and leftmost position available at that instant. The 2D strip packing solution is presented in Figure 3.8(b). The maximum height is 7 in this solution as well. For the backward transformation T, we apply the transformation function for each $x \in \{0, 1, 2, 3\}$. For every value of x, $y = 2$. The transformed instances corresponding to

each x are shown in Figure 3.8(b) in red below the 2-D strip packing solution. We can observe that both Figure 3.8(a) and 3.8(b) give the same offset assignment solution. Thus this example has established that it is indeed possible to use the 2D strip packing problem approaches for solving the offset assignment problem.

3.7.2 Performance Bound for SOAP using BL

In literature the BL algorithm has been shown to have an approximation ratio of 3 [4]. For the signal offset assignment problem we show that by using BL we obtain a performance bound of 2 when the signals have periods that form a geometric series.

To show that Offset Assignment using BL has an approximation ratio of 2 :

Lemma 3.7. *The optimal height should be greater than or equal to the average height of all the slots in a BL instance.*

$$H^* \geq \frac{1}{W}\Sigma_j H_j \tag{3.7}$$

Lemma 3.8. *The optimal height is at least greater than or equal to the maximum height item,* $H^* \geq g_{max}$.

Let $j, 1 \leq j \leq W$ (W is the maximum width for the BL instance) denote the slot in a BL instance (which are transformed to an OA instance using the transformation function). Let H_j denote the height of each slot. Let m_i (with height g_i) denote the last item added to the given BL instance at slot $j = s$. H_s is the maximum height given by the BL solution. As BL places each item on the leftmost bottom most position, therefore the height before the

last assignment is less than or equal to the height of all other slots:

$$H_s - g_i \leq H_j \tag{3.8}$$

Sum the above inequality for all $1 \leq j \leq W$ and divide by W.

$$H_s - g_i \leq \frac{1}{W}\Sigma_j H_j \tag{3.9}$$

From Lemma 2,

$$H_s - g_i \leq H^* \tag{3.10}$$

From Equation 3.10 and Lemma 3.8, we can say that:

$$H_s = (H_s - g_i) + g_i \leq 2 * H^* \tag{3.11}$$

3.7.3 SOAP Using 2D Strip Packing Framework (2DSPF)

Figure 3.9 presents a flowchart to demonstrate the proposed framework of applying 2D strip packing approach for offset assignment in signals.

As discussed in Section 3.7.1 in order to apply SP for offset assignment we need to have signals which have periods that form a geometric series with the common ratio being a positive integer. However in practice this may not always be the case. Therefore, we start the offset assignment using BL (Figure 3.9) by first partitioning the signal periods in the frame such that each group (in a partition) contains elements that form a geometric series. We call such a partition as a *valid partition* for SP. Since there can be several possibilities of partitioning the signal periods, we consider all possible partitions of the given periods

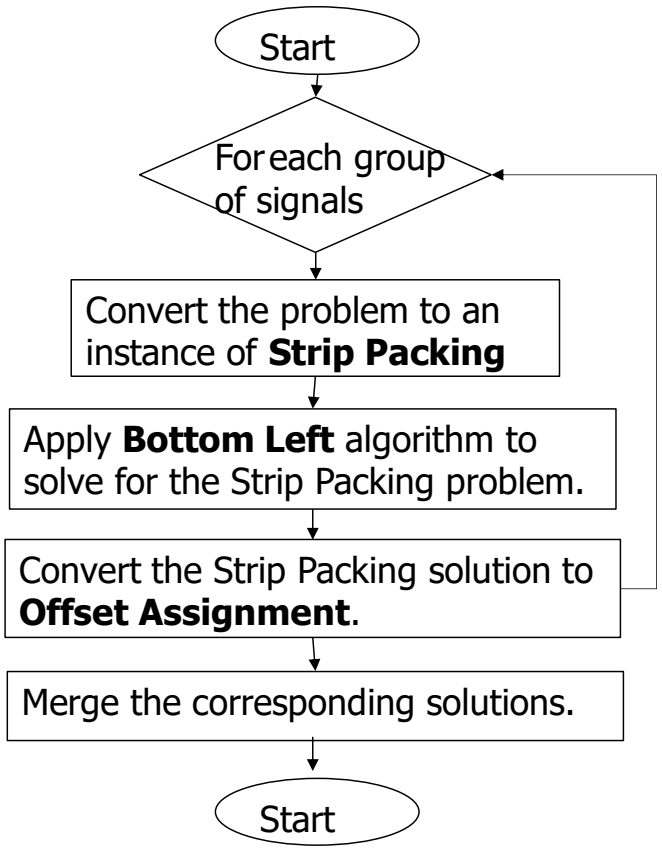


Figure 3.9: Framework Frame Packing in CAN-FD : Offset Assignment using 2D Strip Packing

in a frame. We discard the ones that are not *valid*. For each group in a partition we first transform the OA instance to a SP instance. Next we apply the BL approach for SP and apply the backward transformation to obtain a solution to the OA instance. For each partition we merge the solutions corresponding to all the groups (of that partition). Finally we find solutions of all partitions and choose the minimum solution (corresponding to the frame payload size) from all the partitions. If there are G groups, then by using the 2DSPF we can get a performance bound of $2 * G$.

Example of a *valid* partition Given a set of signals with period $\{1, 2, 5, 10\}$ the *valid* partitions are: (i) $\{1, 2\}, \{5, 10\}$ and (ii)$\{1, 5\}, \{2, 10\}$. Note that partitions like $\{1, 10\}, \{2, 5\}$

or $\{1, 2, 5\}, \{10\}$ are not *valid*.

This framework is then used by us in the greedy frame packing algorithm proposed in our prior work [32]. In the decision making step when the assignment of a signal to an existing frame or a new frame is made, we apply the offset assignment scheme using an SP approach.

3.8 Experimental Results

In this section, we describe the experimental results obtained by applying our proposed GAF for SOAP (with algorithms $\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{A}_3$ for MMP as detailed in Section 3.4) for frame packing in CAN-FD. We use synthetic systems as well as an industrial case study for the evaluation.

The synthetic systems are generated according to the guidelines for real-world automotive benchmarks [39], with minor modifications. Specifically, we redistributed the share of signals with size larger than 64 bytes to the bin "33-64 bytes" (for this work we only consider signals with size up to 64 bytes as the CAN-FD frame payload size is limited to 64 bytes), and the share of signals sent by engine control tasks to those with periods between 1 and 20 ms (as we do not consider the signals with angle-synchronous periods). The distribution of signal periods and payload sizes is given in Table 3.1. In all our systems, we take the deadline of any signal to be equal to its period. We generated 1000 systems for each experiment and averaged the performance parameters over all the systems. For all experiments, we generated systems with a single CAN-FD bus that connects 3 ECUs, and the number of signals are as follows: 50, 80, 100, 120, 150, 200 and 300. The arbitration and data bit rates are set to 500 kbit/sec and 2 Mbit/sec, respectively. We use IBM's CPLEX as the ILP solver [29].

Period (ms)	**Share**	**Size** (Bytes)	**Share**
1	4%	1	35%
2	3%	2	49%
5	3%	4	13%
10	31%	5-8	0.8%
20	31%	9-16	1.3%
50	3%	17-32	0.5%
100	20%	33-64	0.4%
200	1%		
1000	4%		

Table 3.1: Signal parameters and their distribution

3.8.1 Comparison on Bandwidth Utilization: GAF vs No Offset

Since there is no prior work on offset assignment for CAN-FD frame packing, we compare our proposed approaches against the frame packing algorithm in [32] which does not perform offset assignment. Figure 3.10 presents the bandwidth utilization comparison for frame packing using SOAP framework with $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ and no offset assignment. We observe from the plot that offset assignment ($\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$) provides a steadily increasing improvement in bandwidth utilization for frame packing in CAN-FD. In the case of 300 signals the improvement in bandwidth utilization is about 10.54%. We also observe that the bandwidth utilization observed with the offset assignment approaches (in $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$) is almost similar. This shows that in terms of bandwidth utilization, the performance of the SOAP framework with polynomial time heuristics $\mathcal{A}_1$ and $\mathcal{A}_2$ is nearly the same as with $\mathcal{A}_3$ (ILP).

3.8.2 Comparison on System Schedulability: GAF vs No Offset

In addition to bandwidth utilization, we also analyzed the schedulability of the packed CAN-FD frames as described in Section 3.5. In Figure 3.11 we plot the percentage of schedulable and unchedulable systems for each signal size using GAF for SOAP with algorithms $\mathcal{A}_1$,

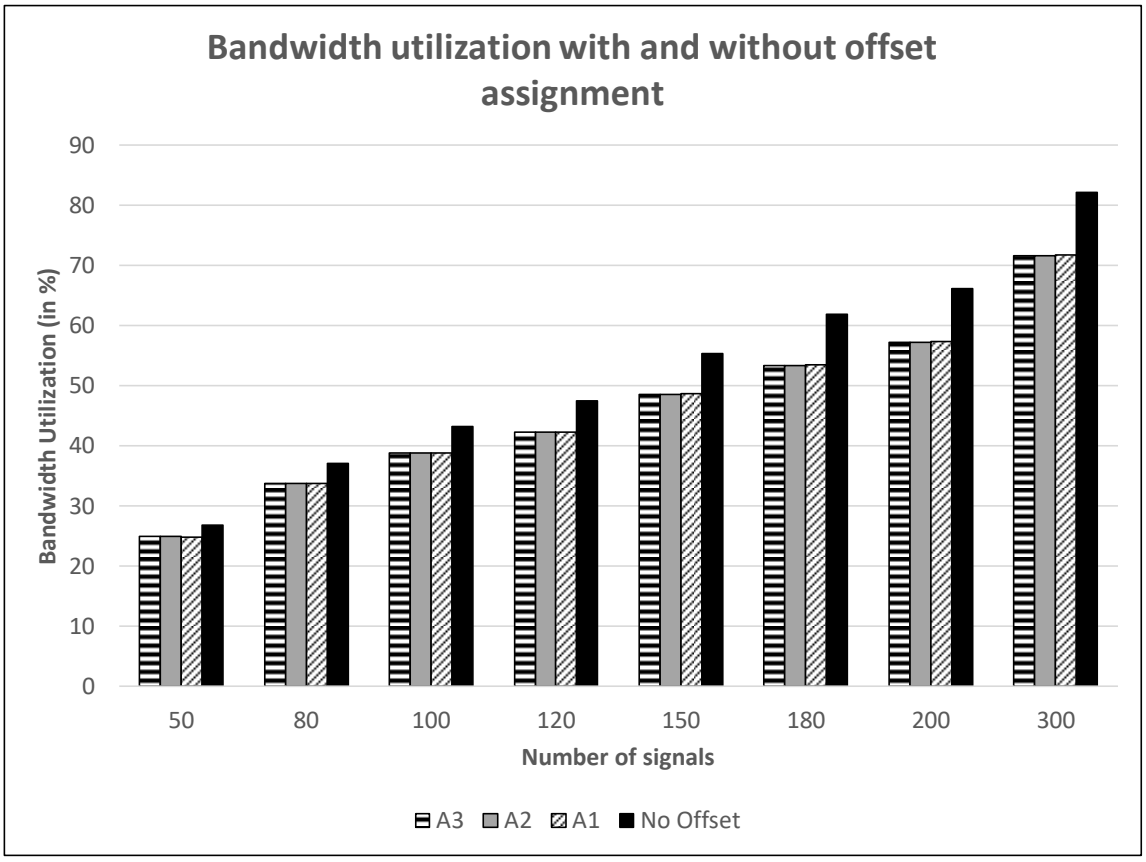


Figure 3.10: Average bandwidth utilization comparison using offset assignment versus no offset assignment.

$\mathcal{A}_2$, $\mathcal{A}_3$ and compare them against simple packing (i.e., no offset assignment). We observe that for small numbers of signals, the performance of the three schemes with respect to schedulability is similar. However, for larger numbers of signals, the offset assignment scheme is able to increase the number of schedulable systems by about 5.4%. This is due to the fact that by appropriately distributing the signals across different instances of a frame, the maximum occupancy of the frame instances is reduced thereby reducing the frame payload size. (Recall from Section 3.2 that the frame payload is taken as the maximum payload of all its instances.) The decrease in frame payload size also reduces the response time of the frame and thus improves schedulability.

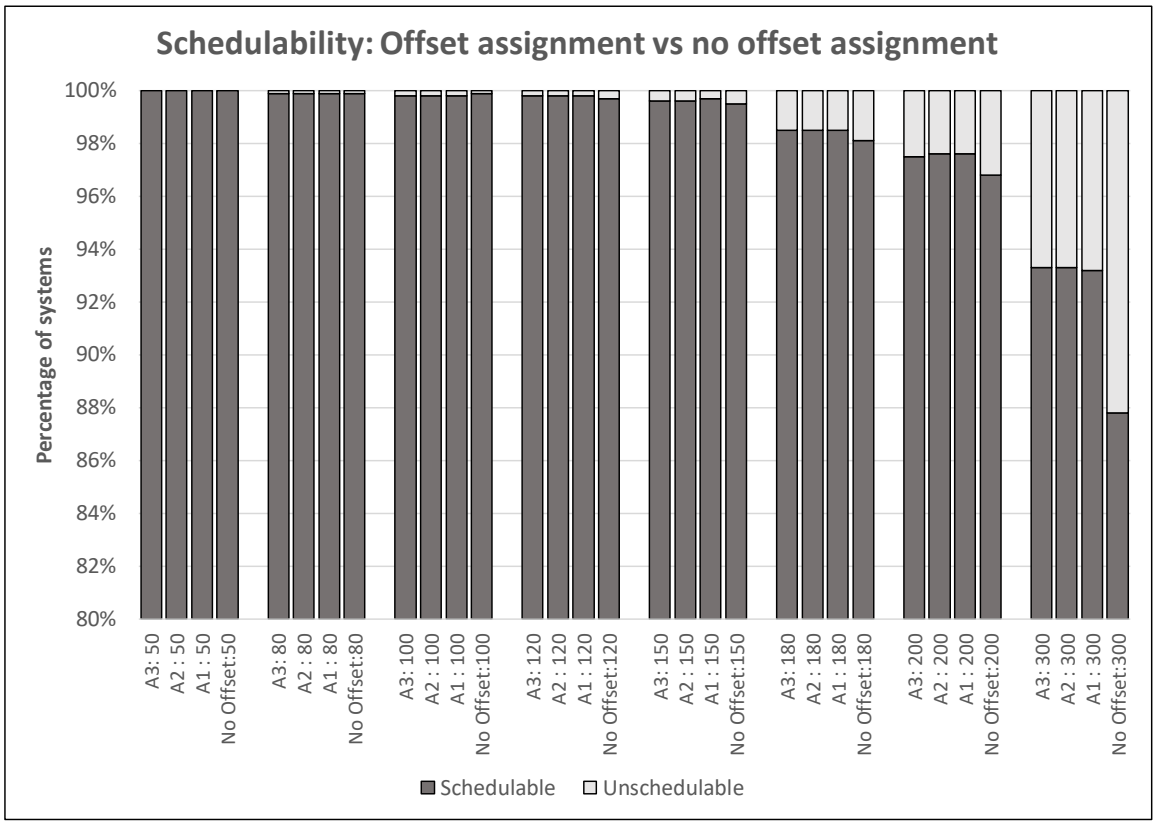


Figure 3.11: Schedulability comparison with and without offset assignment.

3.8.3 An automotive case study: GAF vs No Offset

We also applied the aforementioned frame packing algorithm with our GAF for SOAP using algorithms $\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{A}_3$ on a real automotive case study. In the data set associated with this case study, there are 3429 signals over a CAN-FD bus shared by 17 ECUs. This case study has systems which are broadly in the domains of chassis electronics, body control, adaptive cruise control, and active safety. Figure 3.12 shows the bandwidth utilization for this case study with and without offset assignment. From the figure, it is seen that offset assignment reduces the bandwidth utilization from 29% to about 26.7%. There are several reasons why we see a modest improvement in bandwidth utilization in this case study:

First, the CAN bus is under-utilized, since the simple frame packing scheme (without offset assignment) itself provides a low bandwidth utilization of about 29%. Even so, our offset assignment further reduces the bandwidth utilization by about 2.3%, which is a relative improvement of about 8% considering the original low bandwidth utilization of 29%. In the automotive domain this could mean a potential extension of the architecture's lifetime by several product cycles. Second, the signal set in the case study has a large share (about 69%) with the same period (1000 ms). Therefore, the application of SOAP is restricted in this case. Finally, about 60% of the signals are of size 2 bits or less. As a result, the reduction in the occupancy of a frame resulting from offset assignment is rather small; and due to the discontinuity in CAN-FD frame sizes, a small change in occupancy has very little effect on the size of a frame.

3.8.4 Comparison on Bandwidth Utilization: Improved Heuristics vs GAF

In this experiment we compare the Interchange, Greedy Offset Assignment (GOA), $\mathcal{A}_3$ and No Offset Assignment approaches for frame packing. Figure 3.13 presents the corresponding bandwidth utilization for them. We used the same experimental setup as discussed above. We choose $\mathcal{A}_3$ to represent GAF (since all the three approaches, $\mathcal{A}_3, \mathcal{A}_2$ and $\mathcal{A}_1$ for GAF showed similar results in terms of bandwidth utilization). We observe that GOA provides some improvement over $\mathcal{A}_3$ and Interchange further improves GOA. The improvement in terms of bandwidth utilization over $\mathcal{A}_3$ was observed to be 1.24% by Interchange. The overall improvement of offset assignment compared to no offset assignment by using GOA and Interchange is about 11.477%. We also applied the improved heuristics on the automotive case study and observed a slight improvement of 0.1% in bandwidth utilization as compared to GAF.

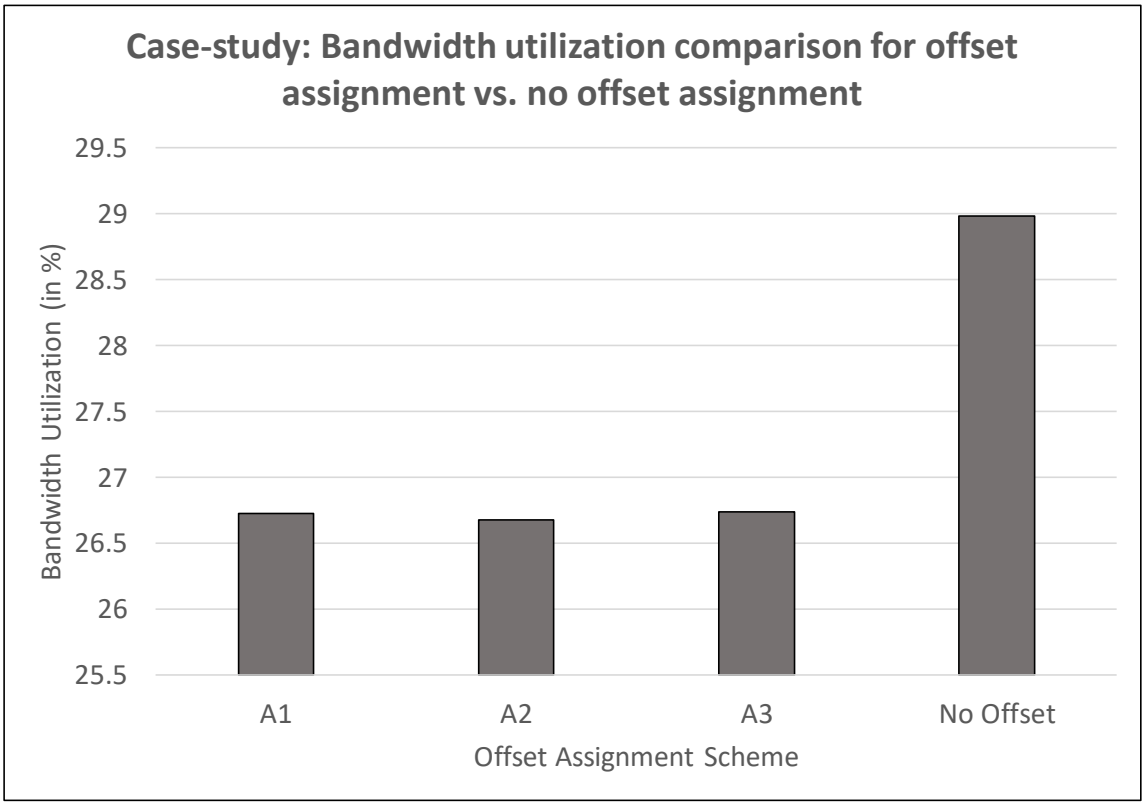


Figure 3.12: Bandwidth utilization comparison for frame packing for the real automotive case study with and without offset assignment.

3.8.5 Comparison on Bandwidth Utilization: 2DSPF vs GAF

In order to test the performance (in terms of bandwidth utilization) of the proposed framework 2DSPF for signal-to-frame packing we perform experiments with different systems as summarized in Table 3.2. We generate synthetic systems using three different rules and the signal sizes range from 50, 80, 100, 120, 150, 180 and 200.

The first three entries in Table 3.2 are experiments performed using synthetic systems generated using the automotive benchmark rules in [39]. We varied the number of ECUs in these experiments as 3, 5 and 10 per system. We compare the average bandwidth utilization for

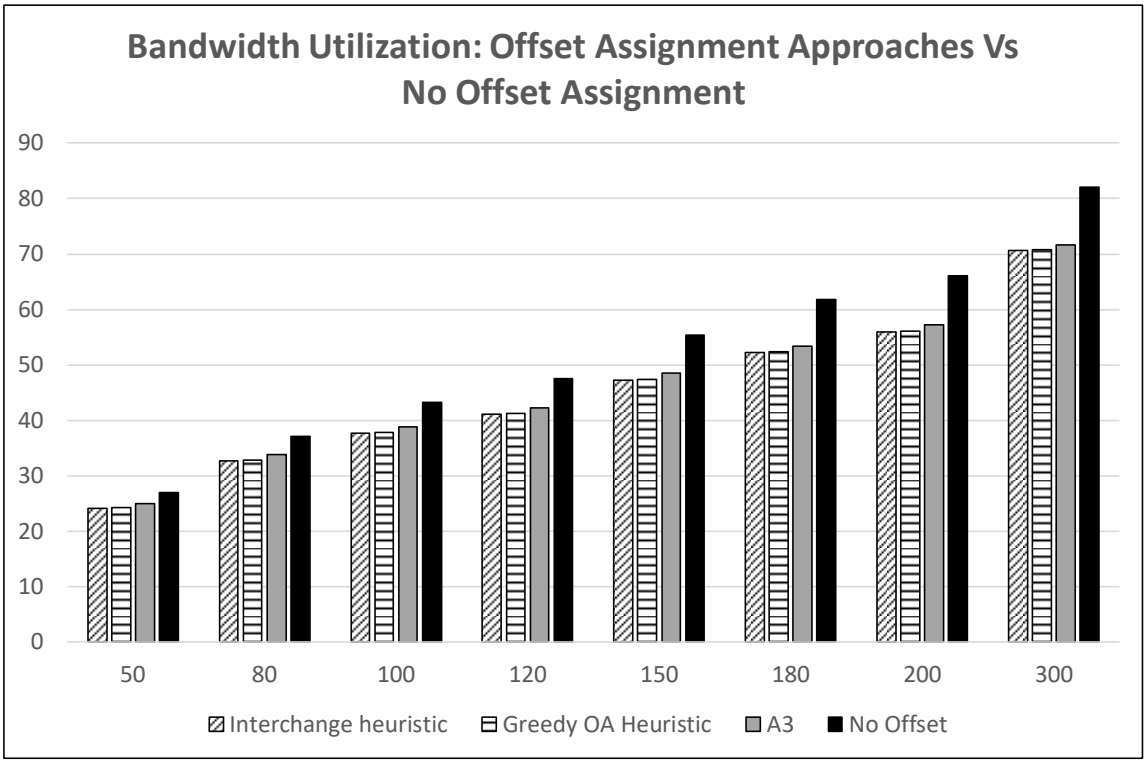


Figure 3.13: Comparison of bandwidth utilization for frame packing using improved offset assignment approaches.

Synthetic Systems Generation Scheme	No. of ECUs Per System	Improvement in BU
Automotive Benchmark [39]	3	0.40%
Automotive Benchmark [39]	5	0.86%
Automotive Benchmark [39]	10	1.39%
Random Systems [76] (Harmonic Periods)	10	1.69%
Random Systems (Periods in GP)	10	1.10%
Automotive Case Study (GM)	17	0.1%

Table 3.2: Bandwidth Utilization Improvement in Frame Packing Using BL for OA

all the systems corresponding to the BL, GAF and No Offset methods. We observe about 0.4%, 0.86% and 1.39% bandwidth utilization improvement in the three experiments.

For the next experiment we generate systems in a different way than the previous experiments. We follow the generation scheme used in [76]. In these systems the periods of the

signals are generated by the product of one to three factors, each randomly drawn from three harmonic sets (2, 4), (6, 12), (5,10). The number of signals range from 50 to 200 and we have 10 ECUs per system. We observe an improvement of about 1.69% in bandwidth utilization by using BL as compared to GAF.

We also performed an experiment on a set of systems having periods in a geometric series. We observe an improvement of about 1.10% in bandwidth utilization by using BL as compared to GAF. We apply our approach on the GM case study and observed a slight improvement of about 0.1% in bandwidth utilization compared to GAF. From these experiments with the 2DSPF we observed a modest improvement in bandwidth utilization, however we plan to perform more experiments to analyze the benefit of our proposed framework.

3.9 Summary

In this chapter, we motivate and formulate the signal offset assignment problem (SOAP) in the context of CAN-FD frame packing, where the goal is to minimize the bus bandwidth utilization by appropriately assigning signals into frame instances. Our contributions are as follows:

- We are the first to motivate, solve and apply SOAP for signal-to-frame packing in CAN-FD. We prove that SOAP is, in general, strongly NP-complete by a reduction from the 3-Partition problem. This rules out the existence of pseudo-polynomial algorithms for SOAP, under the standard assumption that $\mathbf{P} \neq \mathbf{NP}$ [18].

- We propose a **general approximation framework** (GAF) for SOAP that can use any approximation algorithm for the makespan minimization problem (MMP) for multiprocessor systems. We prove that GAF provides a worst-case performance guarantee

of at most ρK, where K is the number of distinct signal periods in the frame and ρ is the performance guarantee of the algorithm for MMP used in GAF. Since several approximation algorithms with ρ close to 1 are known for MMP (e.g., [27, 35]), the performance guarantee provided by GAF is close to K.

- We present experimental results to show that assigning signal offsets in the frame packing step achieves 10.54% improvement in bandwidth utilization over the baseline approach with no offset assignment. We also show improved schedulability from offset assignment compared to the case without offset assignment.

- We propose a new transmission scheme for the AUTOSAR standard [3] for frames that use SOAP. By using this scheme the overhead on the frame payload is lower than in the conventional transmission scheme in AUTOSAR.

- We further improve the offset assignment approach and propose two more heuristics: 1) Greedy Offset Assignment and 2) Interchange. We present experimental results which show that by using these improved approaches for offset assignment we can achieve an improvement of about 11.477% in bandwidth utilization over the baseline approach with no offset assignment.

- We extend our work on SOAP by applying 2D strip packing approach to obtain a better offset assignment to signals for an improvement in frame packing. We propose and develop 2DSPF (2 Dimensional Strip Packing Framework) for frame packing in CAN-FD and present experiments to show it benefits. We prove a performance bound of $2 * G$ for this framework, where G is the number of valid groups of periods (in a frame). We also observe about 1.7% improvement in bandwidth utilization over GAF. We aim to do more experiments for this work which shall be the major focus of our future work.

Chapter 4

Design Space Exploration for TTE-based Architecture of Automotive Systems

4.1 Introduction

In this chapter, we consider the problem of design space exploration with a deterministic architecture based on Time-Triggered Ethernet (TTE). Specifically, we consider the mapping of the given task model onto such a target architecture, as well as the scheduling of the signals communicated among the tasks. The goal is to satisfy the schedulability of tasks and signals, and the end-to-end latency constraints of the critical paths as specified by the designer. Here, we consider task mapping and signal scheduling problems simultaneously because the task mapping on the architecture has a direct effect on the communication scheduling and in turn on the end-to-end latency of the critical path(s). This is a first attempt of its kind, to the best of our knowledge, particularly on the target architecture of TTE.

4.1.1 Related Work

Since the problem discussed here involves both mapping and TTE scheduling, we shall give a brief description of the related work in both the areas. In [30], there is a similar mapping problem defined and its solution is presented in the form of a heuristic. However, the solution does not consider whether the timing constraints of the software model are satisfied after mapping to the architecture model. In order to do that we need to consider the characteristics and properties of the underlying communication protocol on the architecture. In addition to this there are a number of problems in literature on the mapping ([51], [6]) and scheduling of tasks in embedded systems. In [75] a task mapping and scheduling heuristic using an ILP formulation for digital signal processing applications is presented on multi-core architectures. [62] use genetic algorithms to solve the task mapping and scheduling problem and show that their algorithm is fast and their results are near optimal. Similarly [42],[82] present task mapping problems in the domain of embedded systems with objectives like throughput optimization and fast completion time respectively. However none of these work address the problems of task mapping and task communication scheduling together for any network.

4.1.2 State-of-the-art on scheduling of TTE and other networks

The schedule generation of TTE primarily involves the creation of a scheduling table for the TT frames, which is statically generated. The RC frames are event triggered and as such, are integrated in the schedule on their arrival times which are not fixed. In [65] the author uses the concept of "schedule porosity" which leaves blank slots for RC frames while scheduling the TT frames, in order to integrate RC traffic with TT traffic, such that the jitter and latency of RC frames is reasonably reduced. He also gives a latency and jitter analysis of rate constrained traffic in this work. The same author proposes an SMT based

synthesis of TT schedule in [64], however it considers only TT messages and ignores RC traffic. The authors in [80] give another approach to compute the worst-case latency of RC traffic in TTE and to improve it for a better schedulability. In [66], the authors propose a Tabu search based heuristic for obtaining a scheduling solution for the TT frames. They also consider the worst-case delay for RC frames using an analysis from [64] and attempt to reduce the worst-case latencies of RC frames due to the TT traffic in the cluster. This analysis is then used to figure out if the RC frames would meet their respective deadlines.

In addition to TTE, we also give a brief account of problems based on schedulability analysis over other time triggered networks like FlexRay, etc. In [78], an optimization of the task and signal schedule synthesis has been proposed in the form of a mixed integer linear programming problem. Their solution is addressed for the static segment of a FlexRay. In [24] the authors present solutions for FlexRay scheduling for synchronous reactive models. They preserve the semantics of the models by making provisions for deterministic delays on the communication network. They further compare for performance and schedulability and thus present an optimization procedure to minimize loss of performance due to the added delays. For distributed embedded systems with mixed criticality applications (TT and event triggered), [56] gives a scheduling and timing analysis approach. [54] presents synthesis of process and message scheduling over Time Triggered Protocol. They present a schedulability analysis for the fixed priority processes over TTP architecture and propose four scheduling heuristic approaches for message scheduling. In [57], the authors give a timing analysis of tasks and message schedulability (in both the static and dynamic segment) over the FlexRay communication network. These works give a good idea of the issues in solving problems in the same domain, however they cannot be directly used in our case.

Brief comparison of TTE with other protocols: TTE uses a scheduling protocol which is different from other TDMA schemes such as TT-CAN or FlexRay where each cycle consists

of time slots of different sizes for transmission of the signals. For example in FlexRay a cycle consists of a static segment (for time triggered), dynamic segment (for event triggered), symbol window and network idle time [8]. Each node gets a turn to transmit on the bus in dedicated slots over the static and dynamic segments. TTE on the other hand uses a switched network instead of a shared bus system. The TTEthernet pre-emptive switch is capable of identifying TT frames (from the frame identifier) and thus blocks/pre-empts all the other frames to free the outgoing ports for the TT transmission [49]. The RC frames are then sent over the network when there is no TT traffic. This makes scheduling over TTE a little different from other TDMA protocols. Also the latency in TTE is caused due to switching delays [63]. The switch delay is constant and has negligble error for TT frames. In the case of FlexRay, since it is a wired network the latency depends on the length of the wire (from sender to receiver), the propagation delay of the wire and the delay for sending and receiving at the bus drivers [63]. Thus we cannot directly use any work relevant for other protocols directly for solving our problem, and therefore we propose our heuristic as a solution.

4.2 Definitions

***Task Model*:** The task model consists of a directed graph, called a task graph, as shown in Fig. 4.1(a). In the figure, the nodes denote the tasks and the edges represent the signals communicated among tasks. The set of tasks is given by $T = \{\tau_j | j = 1, .., |T|\}$ where each task is defined as a tuple $\tau_j = (o_j, w_j, p_j, d_j, \pi_j)$. Here, o_j gives the offset of the task. The worst case execution time (WCET) of the task is given by w_j. The period of the task is p_j , where $0 \leq o_j < p_j$. The task deadline is denoted as $d_j$which is assumed to be constrained, i.e., $d_j \leq p_j$. The arrival time of the k^{th} instance of the task is given by

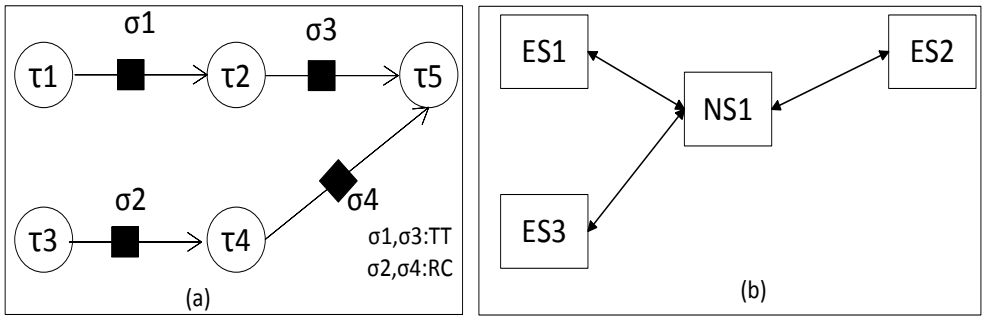


Figure 4.1: (a) Task Model (b) TTE Cluster Architecture Model

$a_{\tau_j,k} = o_j + (k-1) \times p_j$. A task may consume one or multiple input signals at the beginning of its execution and produce one or multiple output signals at the end of its execution. Each task τ_j is assigned with a fixed priority π_j (the higher the number, the higher the priority), and we use $hp(j) = \{\tau_i : \pi_i > \pi_j\}$ to denote the set of higher priority tasks for τ_j. The utilization of a task τ_j is given as $U_j = w_j/p_j$.

In the task graph, the black squares on the edges represent a default buffer for each signal which is used to store the signal before its destination task consumes it. The signals are sent from a source task to a destination task. The set of signals is given by $S = \{\sigma_i | i = 1, ..., |S|\}$ where $\sigma_i = (\tau_i^o, \tau_i^t, p_i, a_i, d_i, l_i)$. Here each signal σ_i is represented as a tuple which contains the source task of the signal τ_i^o, the destination task of the signal τ_i^t, its period p_i, arrival time a_i, deadline d_i and size l_i. The period of the signal is equal to that of its source task. The arrival time of a signal refers to the time when it is produced by the source task i.e., the finish time of its source task (Refer Section 4.5 Step 4). The deadline refers to the maximum time interval between the arrival of the signal and the time it reaches the destination tasks. The size refers to the length of the signal in bytes. If the signal does not reach the destination task within its deadline, then it is lost. The edges also present a precedence constraint with respect to the execution order of the sending (source) task and the receiving (destination) task. The example task model in Fig. 4.1(a) has five tasks,

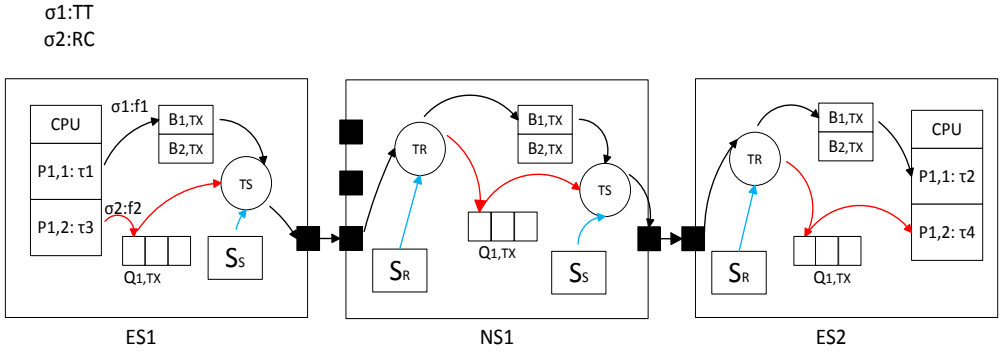


Figure 4.2: A detailed architecture model with an example of frame transmission.

$T = \{\tau_1, ..., \tau_5\}$ and four signals $S = \{\sigma_1, ..., \sigma_4\}$ defined.

***Architecture Model*:** The TTE network is composed of a set of c clusters given by $C = \{C_1, ..., C_c\}$. Each cluster C_k consists of a set of computation units called the end systems (ESes), given by $ES_{C_k} = \{ES_1, ..., ES_N\}$ where N is the number of ESes in each cluster and a set of network switches $NS = \{NS_1, ..., NS_t\}$. The end systems are connected to the network switches by dataflow links, $DL = \{dl_1, ...dl_s\}$. The architecture model in Fig. 4.1(b) gives a representation of a small TTE cluster architecture. In this example we have an architecture model with three ESes, $\{ES_1, ES_2, ES_3\}$ and one NS, $\{NS_1\}$.

The transmission of signals occurs over the TTE architecture after they are packed into frames. The set of frames is denoted as $F = \{f_i | i = 1, .., |F|\}$. A frame $f_j \in F$ is completely specified by the period p_j, size l_j (period, size of the signal that it packs, respectively) and offset θ_j (assigned by the TT-scheduler for each dataflow link [37]). The arrival time of f_j is taken as $A_j = a_j + \theta_j$. The length of the frame in time units can be computed as $g_j = l_j/V$, where V denotes the speed of the dataflow links. All dataflow links are assumed to have the same speed.

Fig. 4.2 presents the architecture of a TTE cluster in detail where the tasks in Fig. 4.1(a) are mapped to the ESes in Fig. 4.1(b) and describes the transmission of both TT and RC

frames in TTE [66]. Task τ_1 produces the signal σ_1 after its execution and packs it into TT frame f_1 and task τ_3 produces σ_2 and packs it into RC frame f_2. The frame f_1 is stored in a dedicated buffer $B_{1,Tx}$(each TT signal from ES_1 has a buffer). The frame f_2 is stored in queue $Q_{1,Tx}$ (each RC VL has a queue). The TT schedule is pre-determined (computed as part of our algorithm) and hence stored in the ESes and the NS, and it guides TS (task sender) on ES_1 when to send f_1 to NS_1. The RC frame f_2 is sent by TS when there is no TT traffic. The path in black shows the transmission of the TT frame f_1 and the one in red shows the transmission of RC frame f_2. There are different mechanisms and tasks for checking the integrity and for sending and receiving of TT and RC frames, however we do not discuss them here. At NS_1 the TT and RC frames are stored again in $B_{1,Tx}$ and $Q_{1,Tx}$ respectively and further transmitted to ES_2 following the same protocol. After reaching the destination (ES_2) TT (f_1) and RC (f_2) frames are stored in $B_{1,Tx}$ and $Q_{1,Tx}$ respectively and are consumed by τ_2 and τ_4 when they get activated respectively.

As evident from the figure, TTE combines the standard Ethernet traffic and the TT traffic with an overhead of a TTE controller. The network switch, NS is actually responsible for this integration as well as for the clock synchronization of the cluster. It also ensures the predictable transmission of time-triggered traffic and flexible transmission of event triggered traffic.

***Scheduling*:** A scheduling solution is an assignment of the frames (containing the signals) to the time windows available for transmission on the dataflow link. Thus for each frame f_i on a dataflow link dl_j, we need to find a start time s_i denoting when it is scheduled to start transmission on dl_i and a finish time e_i denoting when it finishes transmission. The hyperperiod (denoted as H) is taken as the LCM of the period of the frames.

***Critical Path*:** A path $P_{i,j}$ is defined as a sequence of tasks, $P_{i,j} = [\tau_i, ..., \tau_j]$ such that any two consecutive tasks have a common edge, for example $P_{1,5} = [\tau_1, \tau_2, \tau_5]$ in Fig. 4.1(a). The

Table 4.1: Definition of all Notations

$T = \{\tau_j \mid j = 1, .., \mid T \mid\}$	Set of Tasks
$\tau_j = (o_j, w_j, p_j, d_j, \pi_j)$	Task tuple
$S = \{\sigma_i \mid i = 1, ..., \mid S \mid\}$	Set of Signals
$\sigma_i = (\tau_i^o, \tau_i^t, p_i, a_i, d_i, l_i)$	Signal tuple
$C = \{C_1, ..., C_c\}$	Set of clusters
$ES_{C_k} = \{ES_1, ..., ES_N\}$	Set of ESes in cluster C_k
$F = \{f_i \mid i = 1, .., \mid F \mid\}$	Set of Frames
$P_{i,j} = [\tau_i, ..., \tau_j]$	Path
o, p, d, a, π	Offset, Period, Deadline, Arrival time, Priority
w, q, R	WCET, Finish time, Response time
l, V	Length of signal, Speed of dataflow link
$g = l/V$	Transmission time of signal
$U = w/p$	Utilization of task on ES

end-to-end latency of a path can be computed as the time interval between the arrival of an instance of task τ_i and the completion of the instance of task τ_j whose output is dependent on the output of τ_i. We are given a set of critical paths $CP_l \in CP = \{CP_1, ..., CP_L\}$ where $CP_l = [\tau_i, ..., \tau_j]$, and L is the number of critical paths. For convenience, we refer to all the tasks and signals in the critical path as "objects". If there is a data dependency between any two objects in the critical path, the successive objects must start only after all the predecessors complete. We define q and R as the finish time and response time of an object respectively in the critical path.

Table 4.1 gives all the terms defined in the problem so far in order to facilitate the reader for reference.

4.3 Problem Formulation and Algorithm Overview

Given a task model $\mathcal{T} = (T, S)$ where T and S are the task and signal sets respectively, and an architecture $\mathcal{A}$, as specified in Section 4.2, we need to map the task model onto the architecture such that the mapped tasks are schedulable on the corresponding ESes. Once the tasks are mapped we derive the schedule for signals that are communicated among the tasks on the TTE network. We report a successful solution if we have a feasible schedule and the end-to-end latency constraints of the critical paths are met. In addition to that, our mapping solution ensures that the signal traffic over the network is minimized.

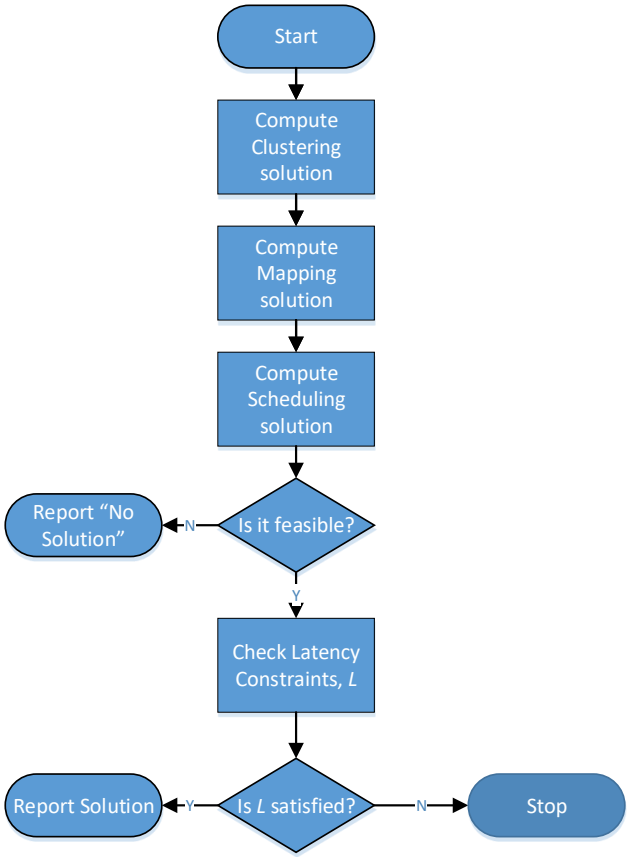


Figure 4.3: Flow-chart for the proposed heuristic

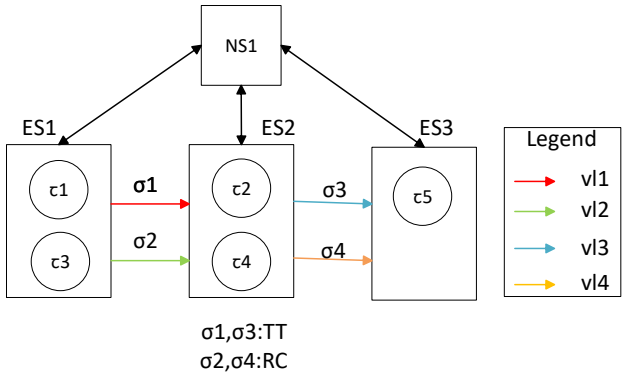


Figure 4.4: Mapping of the task model in Fig. 4.1 on the TTE cluster.

Fig. 4.3 represents a flowchart of the heuristic, which starts with mapping, that can be defined as a function $m : T \rightarrow ES$ that maps the tasks in T to the end systems ES. However in the context of our problem mapping is a two step process:

1) *Clustering the tasks*: A clustering function is defined as $g : T \rightarrow C$ and in our case it is formulated on the basis minimizing the inter-cluster signal traffic. These criteria could include the number and frequency of signals between two tasks, the tasks belonging to the same critical level, tasks belonging to the same sub-system, etc. We group the tasks into clusters by minimizing the inter-cluster signal traffic.

2) *Mapping the tasks onto the ESes*: After the tasks are grouped into the given clusters, we take each cluster and map its tasks onto the ESes. The subset of tasks assigned to a cluster C_k is given by $T_{C_k} \in 2^T$. We map the tasks by optimizing the inter-ES signal traffic and enforcing schedulability constraints on each ES. Fig. 4.4 gives an example of mapping of tasks in the task model in Fig. 4.1(a) to the architecture in Fig. 4.1(b). Tasks τ_1 and τ_3 are mapped on $ES1$, tasks τ_2 and τ_4 are mapped on $ES2$ and task τ_5 is on $ES3$.

3) *Scheduling*: After the mapping step, we consider the problem of scheduling the signals communicated among the tasks over the TTE network. Tasks mapped to the same processor

communicate using shared memory and we do not model that explicitly here. Using the task graph and the mapping solution we can create a signal list for each dataflow link on a cluster. The signals are packed into frames and for our purpose we assume that each frame f_j contains a single signal σ_j (in this work we do not consider the problem of frame packing). We denote f_{dl_i} as the frame list for the frames to be scheduled for link dl_i. We need to compute the start time s_i and end time e_i for each signal σ_i. As described above, in TTE network the TT frames are given highest priority for transmission In this work we only schedule the TT frames. A scheduling solution is feasible if all the frames meet their respective deadlines. After the mapping and scheduling stages we compute the end-to-end latencies of the given critical paths and verify whether they lie within the deadline requirements set by the designer.

4.4 Complexity of the Problem

We now try to reduce the Job Shop Scheduling problem (JSSP), which is known to be NP-complete [72], to our problem. The JSSP is defined as follows [72]:

Problem 1. There is a finite set J with n jobs, a set O with N operations and a set M of m machines. Each operation $v \in O$ is associated with a job $j \in J$, a machine $m \in M$ on which it gets processed and a processing time $t \in \mathbb{N}$. Also there is a precedence relation, $\rightarrow$ among the elements in O which gives the sequence of scheduling the operations. The problem is to find a start time, s_v for each operation $v \in O$ so that max $s_v + t_v$ is minimized. This is subject to the constraints:

$$s_v \geq 0, \quad \forall v \in O \tag{4.1}$$

$$\text{if } u \rightarrow v, \quad \{u, v\} \in O \text{ then } s_v - s_u \geq t_u \tag{4.2}$$

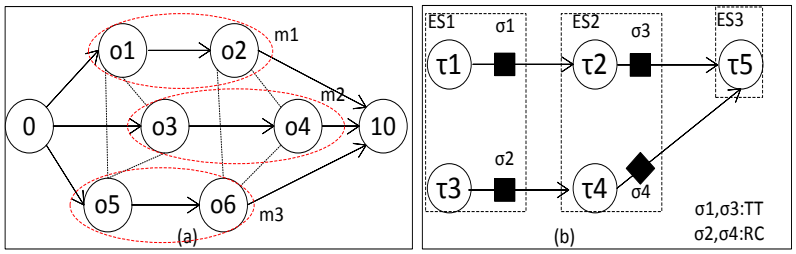


Figure 4.5: (a) Disjunctive graph of 3 Jobs with 2 Operations for JSSP (b) Task graph of MSP

$$\text{if } M_u = M_v, \text{ then } s_v - s_u \geq t_u \bigvee s_u - s_v \geq t_v \tag{4.3}$$

We now compare the JSSP problem to the problem in this paper (hereby referred to as mapping-scheduling problem or MSP). The 'jobs' (J) in JSSP are analogous to the 'tasks' (T) in MSP. A subset of these tasks ($T_{src} \subseteq T$) produces 'signals' (S), where T_{src} denotes the set of source tasks for the signals. The set of signals, S in MSP is analogous to the set of 'operations' (O) in JSSP. Each operation is associated to a job J, similarly in MSP each signal is associated to a source task T_{src}. The goal of JSSP is to assign the jobs to the 'machines' (M) and schedule the operations on the machines such that the time taken to execute the longest chain of operations is minimized. Fig. 4.5(a) presents a disjunctive graph to show an example of a JSSP with 3 jobs with 2 operations each ($\{o_1, ..., o_6\}$). The operations 0 and 10 are fictitious operations that denote the start and end operations respectively. The solid arrows denote the precedence relation between two operations and the dotted lines represent the assignment of the operations to the machines. Operations o_1 and o_2 are mapped to machine 1, o_3 and o_4 to machine 2 and o_5 and o_6 to machine 3. We compare this with the data-flow graph for MSP presented in Fig. 4.5(b), where each task is mapped to an ES (thus mapping the signals to the ES). The problem that MSP needs to address vis-à-vis JSSP is as follows: After mapping the tasks to the ESes, find the start time, s_i of each signal, $\sigma_i \in S$

over the appropriate dataflow link such that,

$$s_i \geq 0 \tag{4.4}$$

The end time e_i for σ_i is given by, $e_i = s_i + g_i$, where g_i is the length of the signal frame in time units. A signal is scheduled only after the previous signal ends, i.e., if σ_j is scheduled later than σ_i then the start time of σ_j cannot begin before the end time of σ_i

$$\text{if } s_i < s_j \text{ then } s_j - s_i \geq g_i \tag{4.5}$$

Also, each signal is scheduled as it arrives and according to its priority as TT or RC. Equations 4.5 and 4.2 are analogous to Equations 4.4 and 4.5 respectively. The objective of the problem is that after a feasible schedule of the signals is achieved, the latency of the critical path given by $CP = [\tau_1, ..., \tau_s]$ should lie within the specified bound($\mathcal{B}$):

$$\tau_{s(CP)} - \tau_{1(CP)} \leq \mathcal{B} \tag{4.6}$$

Here τ_s and τ_1 are the last and first tasks in CP. If we assign $T_{src} = T$ (i.e. each task produces a signal) and CP is taken as the longest path in the dataflow graph, then we can transform MSP to JSSP. This transformation can take place in polynomial time since $T = T_{src} + T'_{src}$ and a longest path L can be represented as a combination of smaller paths in the dataflow graph. Thus, from the above discussion JSSP can be solved in polynomial time if MSP is solved in polynomial time. However, since JSSP is a known NP-complete problem [72], therefore we can claim that MSP is also NP-complete. In order to illustrate the entire section better we tabulate all the different variables used in both the MSP and JSSP in Table 4.2.

Table 4.2: Comparing MSP to JSSP

JSSP	MSP
Set of Jobs: $j \in J$	Set of Tasks: $\tau \in T$
Set of Operations: $v \in O$	Set of Signals: $\sigma \in S$
Start time of v: s_v	Start time of σ_i: s_i
Processing time of v: t_v	Transmission time of σ_i: g_i
Obj: Min(Max $(s_v + t_v)$)	Obj: For path P, $\tau_s - \tau_1 \leq \mathcal{B}$

4.5 Details of the Heuristic

We describe all the steps of the heuristic in detail here:

Step 1: Clustering: We have a set of clusters $C = \{C_1, ..., C_c\}$ and the problem is to group the set of tasks T into these c clusters. The grouping is performed such that the inter-cluster communication is minimized. We use an integer linear programming (ILP) formulation to solve this problem. A binary decision variable, x is defined as follows:

$$x_{i,j} = \begin{cases} 1 & \text{if task } i \text{ is mapped to cluster } j \\ 0 & \text{otherwise} \end{cases}$$

Hence, for every cluster the total utilization of all the ESes over a cluster should be less than or equal to 100%, therefore summing x over each of the tasks gives:

$$\forall j, \quad \sum_i (x_{i,j} \times U_i) \leq N_j \tag{4.7}$$

where N_j is the number of end systems in cluster j. Since each task is mapped to exactly one cluster, for every task, summing x over each of the clusters gives:

$$\forall i, \quad \sum_j x_{i,j} = 1 \tag{4.8}$$

Now, we define a real decision variable z for a signal s between source task u and destination task v and cluster j. This is done in order to relax the solution and have a linear objective function.

$$z_{s,j} = \begin{cases} 1 & \text{if both } u \text{ and } v \text{ for } s \text{ are in cluster } j \\ 0 & \text{otherwise} \end{cases}$$

Thus, we have the following inequalities between x and z.

$$x_{u,j} + x_{v,j} \leq z_{s,j} + 1 \tag{4.9}$$

$$z_{s,j} \leq x_{u,j} \tag{4.10}$$

$$z_{s,j} \leq x_{v,j} \tag{4.11}$$

The objective function is chosen to minimize the inter-cluster signals (or maximize the intra-cluster communication) to facilitate the scheduling step and in turn the end goal, which is end-to-end latency satisfaction. Here, $\{S|\tau_o = u, \tau_t = v\}$ denotes the set of signals whose origin task is u and target task is v.

$$\max \sum_{j \in C} \sum_{\{S|\tau_o = u, \tau_t = v\}} z_{s,j} \tag{4.12}$$

Step 2: Mapping: The mapping step assigns the set of tasks T to the end systems, ES in each cluster, subject to schedulability on the respective ES. In order to check the schedulability of the tasks on the ES we use the request bound function to evaluate the feasibility of fixed priority systems [76]. Response time equation is used as a necessary and sufficient condition to check for schedulability. However for large task systems it becomes too complex to evaluate online. Hence, [76] proposes a novel approach for analyzing the schedulability of periodic tasks for an ES. We first need to find the test points to check the

satisfaction of the schedulability condition (as defined in [76]). We define binary decision variables $y_{i,j}$ for each task i, in the cluster C_k as follows:

$$y_{i,j} = \begin{cases} 1 & \text{if task } i \text{ is mapped to } ES_j \in ES_{C_k} \\ 0 & \text{otherwise} \end{cases}$$

Each task has N such variables defined where N is the number of end systems in the cluster C_k. The following equation gives the request bound function condition for schedulability [76].

$$\bigwedge_{i=1,\ldots,|T|} \bigvee_{t \in B_i} (w_i + \sum_{j \in hp(i)} \left\lceil \frac{t}{p_j} \right\rceil w_j \leq t) \tag{4.13}$$

In (4.13), w_i gives the WCET of the task i and p_j and w_j denote the period and WCET of the task j with priority higher than task i, respectively. Each task has a corresponding set of test points given by B_i. The request bound function gives the response time for each task and this should be less than the value of the test point t. For each task this inequality should be satisfied by at least one of the test points from its corresponding set (hence the OR operation for the test points). It should also hold for all the tasks mapped onto an ES. [76] provides a set of procedures to identify the test points for each task i . Using (4.13) we construct the constraints for mapping, which ensure that the tasks mapped onto the ESes satisfy the schedulability constraints as follows: $\forall ES_m \in ES_{C_k}$

$$\bigwedge_{\tau_i \in T_{C_k}} \bigvee_{t \in B_i} (w_i * y_{i,m}) + \sum_{\tau_j \in hp(i) \in T_{C_k}} (\left\lceil \frac{t}{p_j} \right\rceil w_j * y_{j,m}) \leq t \tag{4.14}$$

We also define a constraint for maximum utilization of an end system: $\forall ES_m \in ES_{C_k}$,

$$\sum_{\tau_i \in T_{C_k}} (y_{i,m} \times U_i) \leq 100\% \tag{4.15}$$

From (4.14) there are $(N_k * |T_{C_k}| * |B_i|)$ constraints for cluster C_k, where N_k is the number of ESes on the cluster, $|T_{C_k}|$ is the number of tasks in cluster C_k and $|B_i|$ is the number of test points for task τ_i. The objective function for this problem is created to minimize the signal traffic on the TTE network (or maximize the intra-ES communication), to facilitate the feasibility of the next step (communication scheduling).

$$\max \sum_{ES_m \in ES_{C_k}} \sum_{\{S|\tau_o=u,\tau_t=v\}} (y_{u,m} * y_{v,m}) \tag{4.16}$$

Such a nonlinear function containing the product of two binary variables can easily be linearized in a similar fashion as [76].

Step 3: Scheduling: In order to derive a scheduling solution for the set of frames denoted by $F = \{f_i | i = 1, .., |S|\}$ corresponding to the set of signals S, we take the list of frames that are scheduled to be sent to a dataflow link dl_j (f_{dl_j}) and then schedule them according to the algorithm given as follows. Also, for each frame we ensure the constraints in (4.17), (4.18) and (4.19) are satisfied, i.e. the frames can be scheduled for transmission only after they arrive, the finish time of a frame is the sum of its start time and its transmission time (i.e. frames are not preempted) and no two frames should have any overlap during transmission.

$$s_i > A_i \tag{4.17}$$

$$e_i = s_i + g_i \tag{4.18}$$

$$\text{if } s_i < s_j, \text{ then } s_j > e_i \tag{4.19}$$

In Algorithm 4 (FindSchedule), a list of frames (*Ready_List*) to be scheduled on a dataflow link dl_j is prepared and sorted according to the arrival times of the frames. We call the Schedule algorithm for scheduling the TT frames. In Algorithm 4.6 starting from the top of

Algorithm 4 Find Schedule

```
1: procedure FindSchedule
2:     Sort(S(ψ_i))
3:     ScheduleList = {}
4:     Schedule(TT)
```

the frame list, a frame f_i is taken. We then find the next available slot (*available_slot*) after the arrival time of the frame A_i, of length at least equal to the length of the given frame, g_i. If such a slot is available we check whether scheduling the frame in this slot will meet the deadline d_i of the frame. If both these conditions are met, frame f_i is scheduled in the available slot. *ScheduleList* is populated with the frame f_i and its sending time s_i and the counter is incremented to go to the next frame in the *Ready_List*. If there is no available slot found or the deadline of the given frame is not met by scheduling it in the available slot then the algorithm reports an infeasible schedule for f_i.

Step 4: Latency Computation: After obtaining a feasible scheduling solution, the final step is to compute the latency of the critical path(s). The computed latency is then compared to the latency constraint given by the designer and if it is satisfied we report the corresponding mapping and scheduling as a valid solution. We are given one or multiple critical path(s) with their latency constraints. For a given critical path, $CP = [\tau_i, ..., \tau_j]$ with signals $\{\sigma_m, ..., \sigma_n\}$, we compute the end-to-end latency using the following equation:

$$q_{snk(CP)} - a_{src(CP)} \tag{4.20}$$

Here, $q_{snk(CP)}$ and $a_{src(CP)}$ denote the finish time of the last object and the arrival time of the first object of CP respectively. In order to compute (4.20) we consider each object from CP and calculate its response time, R and finish time, q. Since the tasks and signals are scheduled periodically, we need to take into account the appropriate instance of the object for

Schedule

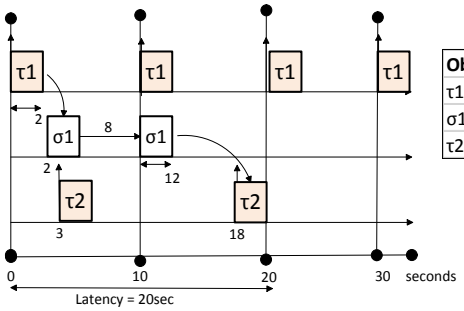
1. Sort the *Ready_List*.
2. **for** $f_i \in Ready\_List$ **do**
   (a) find **next_available_slot**:
   availableSlot = findAvailableSlot(A_i, g_i)
   if(availableSlot is empty) **then**
   if(availableSlot.startTime + $g_i \leq d_i$) **then**
   ScheduleList.insert(f_i, availableSlot.startTime)
   else
   f_i schedule infeasible
   endif
   endif

endfor

Figure 4.6: Schedule Algorithm

latency computation. The successive objects in CP must start only after all the predecessors have finished. For example if a critical path consists of a task τ_i which sends a signal σ_m to task τ_j then, $q_i \leq A_m$ and $q_m \leq a_j$. Fig. 4.7 demonstrates the latency computation for the above example. The hyperperiod of the objects is 30 seconds. We consider the first instance of task τ_1 and after it finishes execution (at 2 sec.) and produces signal σ_1, we consider the scheduling of σ_1. By the time σ_1 is delivered to the destination task τ_2 (at 12 sec.), the first instance of τ_2 is finished and hence we consider the second instance which arrives at 18 seconds. The total latency is the duration from the time τ_1 arrives to the time τ_2 finishes. For any task τ_i, the response time is computed using the response time equation:

$$R_i = w_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{p_j} \right\rceil * w_j \tag{4.21}$$



Object	Period	Offset	WCET/size
τ1	10	0	2
σ1	10	8	2
τ2	15	3	2

Figure 4.7: An example to compute end-to-end latency

w_i is the WCET for the task τ_i and w_j, p_j are the WCET and period for tasks with priority higher than task τ_i. Here the response time R_i appears on both sides of the equation, hence we use the fixed point method to compute the response time for each task. The finish time of a task is given by:

$$q_i = a_{\tau_i} + (l-1) * p_i + R_i \tag{4.22}$$

Here, l is the instance of the task τ_i in the hyperperiod. In the communication schedule, each TT frame has a fixed pre-computed schedule which is repeated every hyperperiod (H). The TTE scheduler assigns offset to each TT frame. In our problem this offset is fixed for each instance of the TT frame. Thus for computing the absolute finish time of a signal we need to find its appropriate instance that is scheduled after the sender task finishes and writes the signal into the buffer. The response time for a signal frame f_i is computed as:

$$R_i = e_i - A_i + TL \tag{4.23}$$

Here A_i denotes the time when frame f_i is ready to be transmitted on the link (taken as the finish time of the source task), and e_i denotes the time when f_i is delivered to the destination task. TL denotes the technical latency introduced by the switch (taken as $5\mu s$ [66]). The

finish time of frame f_i is given by:

$$q_i = A_i + (k-1) * p_i + R_i \tag{4.24}$$

Here, k is the instance of the signal σ_i in the hyperperiod.

4.6 Experimental Results

In this work we attempted to solve the mapping and scheduling problem in automotive design with the end goal of satisfying the end-to-end latency. The clustering and mapping steps minimize the inter-cluster and inter-ES signal communication. This is done in order to facilitate scheduling and reduce the response times of latency paths in general. However, in addition to the main goal, we also minimize signal traffic and signal response times. We have used C++ and IBM's CPLEX for implementing the heuristic algorithm. We present the experimental results on three benchmarks, with 49, 100 and 150 tasks respectively. The first benchmark is a real case-study (for a subsystem) from a well-known automotive industry[1] whereas the other two are synthetically created. Table 4.3 gives the dimensions (number of task, signals, ESes and constraints developed) for the case studies used by us. Table 4.5 presents the results for these three benchmarks where the first column on Scheduling gives the percentage of the signals successfully scheduled (and meeting their deadlines) by the approach, the second column gives the end-to-end latency in milliseconds and the third column gives the time taken by the complete algorithm. The allowed end-to-end latencies for different critical paths are specified by the designer for each case. As evident from the table, all the signals meet their deadlines and all the critical paths satisfy the latency constraints.

[1] An industrial automotive system comprises of several subsystems like chassis, engine control, etc. and employs about 100 ECUs in a complex network spread over upto 10 buses and over 2500 signals [17].

Table 4.3: Testbenches for Experiments

Set	Tasks	Signals	ESes	Constraints
1	49	132	15	105
2	100	264	60	208
3	150	300	72	342

Table 4.4: Mapping for 49 Tasks

Cluster1			Cluster2		
ES1	ES2	ES3	ES1	ES2	ES3
22	0	0	11	16	0

Table 4.5: Results

Set	Scheduling	End-to-endLatency(ms)		Time Taken
	TT	Allowed	Computed	(minutes)
1	100%	Path1: 100	1.76	0.25
2	100%	Path1: 100	77.06	27.28
		Path2: 240	166.4	
3	100%	Path1: 100	37.85	976.36
		Path2: 240	160.88	

As a result of the minimization of local cluster and ES traffic the reported mapping of tasks to clusters and ESes is very compact. An example of such a mapping is presented in Table 4.4, where the numbers in the cells correspond to the number of tasks mapped onto the clusters and the ESes. Thus we see that the algorithm utilizes just 3 of the available 15 ESes for mapping (for task set 49). The heuristic gives a more compact packing of tasks onto the ESes in the case of 100 tasks than the 150 tasks, hence the computed latencies are higher due to higher interference and hence higher response times of tasks.

Fig. 4.8 presents how the algorithm scales for different sets of tasks. From this chart the heuristic gives a solution in about a quarter of a minute for the industrial benchmark. However, as the size of the set increases it is evident from the chart that the time taken to converge to a solution gets larger. For example, with 150 tasks the runtime of the heuristic is over 15 hours. This is mainly due to the use of ILP in the procedure.

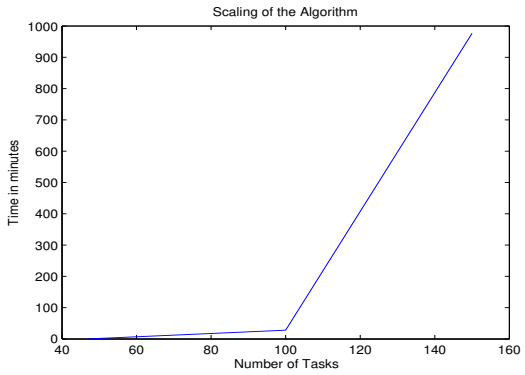


Figure 4.8: Algorithm scaling chart

4.7 Summary

In this chapter we have attempted to solve an important problem of mapping and scheduling for embedded system design of automotive systems. The main goal of the problem is to obtain a solution for mapping the task model onto the architecture and to generate a feasible schedule for the TTsignals communicated by the tasks on the TTE network, such that they satisfy the latency constraints set by the designer. This will ensure that the mapping solution preserves the timing constraints of the task model. We present our heuristic algorithm, its implementation and results on some benchmarks which have been used as our case studies. Our algorithm has four steps, where each step facilitates the optimal solution of the subsequent step. This gives a useful technique to obtain solutions for multi-objective problems like these.

The solution presented in this work is a complete solution for the given problem. This is due to the fact that only reporting a mapping or a scheduling solution maybe optimal in itself but when combined the complete solution may turn out to be infeasible. For example for a particular task mapping solution there may be no feasible schedule for all the signals, and

therefore we should reject that mapping (even if it is optimal) in the design phase. Hence considering scheduling after the mapping gives the mapping solution a completeness so that it can be applied to solve real time problems. Due to the nature of the problem and its complexity it consists of a number of sub-problems such as improvement of schedulability, improvement of clustering methods, achieving the desired end-to-end latency, etc. Hence we wish to continue our work along these lines and make the approach more robust. This also requires more testing on real-time data in order to show the reliability and efficiency of the approach.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this dissertation we discuss some of the design space exploration challenges in automotives. The automotive domain is rapidly evolving with the addition of more and more modern features into it, such as adaptive cruise control, collision avoidance and various other driver assistance systems. As electrical units and software play a major role in the design of automotives nowadays, there are a number of challenges that have sprung up in the process of developing a reliable and efficient system design methodology for automotives.

The focus of this work is on the real time communication protocols such as CAN /CAN-FD and TTE. The major contribution of this dissertation is devoted to the problem of bandwidth optimization over CAN-FD. The motivation of bandwidth optimization is directly linked to the industrial requirement of addition of the growing features in modern automotives as well as the reliable transmission of signals over the network. We present two main contributions in this regard. The first one addresses the multi-domain frame packing problem for CAN-FD which formulates the problem of signal-to-frame packing in a CAN-FD (compared to bin-packing in the literature) system with multiple CAN-FD networks. We propose and develop an optimal (ILP) and several greedy heuristic solutions to this problem with two different models for the central gateway. Secondly we propose a signal offset assignment scheme which improves the frame packing in CAN-FD. We present approximation algorithms and several

heuristics for this problem and show the importance of this technique in saving bandwidth with different experiments and also an industrial case study.

The third contribution in this dissertation focuses on the mapping of a given task model onto an architecture based on time triggered Ethernet (TTE). The objective of this problem is to produce a feasible schedule for signals transmitted by the different tasks such that no signal misses its deadline and optimization of the end-to-end latency of all the critical paths in a sub-system.

5.2 Future Work

Our future work would consist of tackling similar problems in the automotive domain.

1. In Chapter 2 we have solved the multi-domain frame packing problem using "Basic" and "Advanced" gateway models. For the future work we intend to consider a more sophisticated model for the advanced gateway which can decompose several frames and then repack signals from different frames together. This model would require a more careful timing analysis of the arrival times of the frames and also need a buffer system to store the signals temporarily. Another direction for future work is to consider a heterogeneous network with CAN, Ethernet, etc., for the multi-domain system.

2. We have introduced the signal offset assignment problem (SOAP) for frame packing in CAN-FD in Chapter 3. We also proposed a framework for applying 2-D strip packing approaches to obtain an offset assignment to signals in Section 3.7. We present some results in Section 3.8.5 which show a comparison between the proposed framework and our previously developed approach (Generalized Approximation Framework). We observed an improvement in bandwidth utilization of about 1.7%. As part of the

future work we intend to perform more experiments on different systems to completely analyze the performance of the proposed framework.

3. The other open problem is that of having a variable data-length code (DLC) for CAN-FD frames. The underlying idea behind this problem is to have a flexible DLC for a CAN-FD frame which shrinks and expands as per the payload it is carrying. This would intuitively reduce the bandwidth utilization of the CAN-FD bus because the loss incurred due to the packing of signals with different periods would be eliminated.

   In Chapter 3 we have proposed a novel transmission scheme for offset assignment (with an overhead of a single bit) for AUTOSAR. This scheme would also work for the variable DLC case. However the schedulability for the variable DLC case requires the more generalized analysis which follows from the case of multi-frame messages for CAN where each frame has a set of different sub-frames [41].

Bibliography

[1] *Specification of I-PDU Multiplexer V3.2.0 R4.0 Rev 3.*

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

[3] GbR AUTOSAR. Automotive open system architecture. *URL http://www. autosar. org*, 2007.

[4] Brenda S Baker, Edward G Coffman, Jr, and Ronald L Rivest. Orthogonal packings in two dimensions. *SIAM Journal on computing*, 9(4):846–855, 1980.

[5] Unmesh Dutta Bordoloi and Soheil Samii. The frame packing problem for CAN-FD. In *Proceedings of the IEEE 35th IEEE Real-Time Systems Symposium (RTSS)*, pages 284–293, 2014.

[6] Tracy D Braun, HJ Siegal, Noah Beck, Ladislau L Boloni, Muthucumaru Maheswaran, Albert I Reuther, James P Robertson, Mitchell D Theys, Bin Yao, Debra Hensgen, et al. A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing systems. In *Heterogeneous Computing Workshop, 1999.(HCW'99) Proceedings. Eighth*, pages 15–29. IEEE, 1999.

[7] Bernard Chazelle. The bottomn-left bin-packing heuristic: An efficient implementation. *IEEE Transactions on Computers*, (8):697–707, 1983.

[8] FlexRay Consortium et al. Flexray communications system-protocol specification. *Version*, 2(1):198–207, 2005.

[9] Armaghan Darbandi, Sungoh Kwon, and Myung Kyun Kim. Scheduling of time triggered messages in static segment of FlexRay. *International Journal of Software Engineering and its Applications*, 8(6):195–208, 2014.

[10] Robert I. Davis, Alan Burns, Reinder J. Bril, and Johan J. Lukkien. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, 2007.

[11] Robert I Davis, Liliana Cucu-Grosjean, Marko Bertogna, and Alan Burns. A review of priority assignment in real-time systems. *Journal of Systems Architecture*, 65:64–82, 2016.

[12] Marco Di Natale, Haibo Zeng, Paolo Giusto, and Arkadeb Ghosal. *Understanding and using the controller area network communication protocol: theory and practice*. Springer Science & Business Media, 2012.

[13] David Fernández-Baca. Allocating modules to processors in a distributed system. *IEEE Transactions on Software Engineering*, 15(11):1427–1436, 1989.

[14] Felix Fikke. Electric/electronic-architectures: Automating and optimizing communication matrices. 2016.

[15] Greg Finn and Ellis Horowitz. A linear time approximation algorithm for multiprocessor scheduling. *BIT Numerical Mathematics*, 19(3):312–320, 1979.

[16] Donald K. Friesen and Michael A. Langston. Variable sized bin packing. *SIAM Journal on Computing*, 15(1):222–230, 1986.

[17] Simon Fürst. Challenges in the design of automotive software. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, pages 256–258. IEEE, 2010.

[18] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 1979.

[19] Michael R Garey and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Company, San Francisco, CA, 1979.

[20] Ronald L Graham. Bounds for certain multiprocessing anomalies. *Bell Labs Technical Journal*, 45(9):1563–1581, 1966.

[21] Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics*, 17(2):416–429, 1969.

[22] Mathieu Grenier, Lionel Havet, and Nicolas Navet. Pushing the limits of CAN-scheduling frames with offsets provides a major performance boost. In *4th European Congress on Embedded Real Time Software*, 2008.

[23] OSEK Group et al. Osek/vdx operating system specification. *site web: http://www. osek-vdx. org*, 2005.

[24] Gang Han, Marco Di Natale, Haibo Zeng, Xue Liu, and Wenhua Dou. Optimizing the implementation of real-time simulink models onto distributed automotive architectures. *Journal of Systems Architecture*, 59(10):1115–1127, 2013.

[25] Florian Hartwich. CAN with flexible data-rate. In *Proc. 13th International CAN Conference (iCC)*, pages 14:1–14:9. CAN in Automation (CiA), 2012.

[26] Christian Herber, Andre Richter, Thomas Wild, and Andreas Herkersdorf. Real-time capable can to avb ethernet gateway using frame aggregation and scheduling. In *Design, Automation & Test in Europe Conference & Exhibition*, pages 61–66. IEEE, 2015.

[27] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *J. ACM*, 34(1):144–162, Jan. 1987.

[28] Thomas Hogenmller and Burkhard Triess. Cost efficient gateway architectures for deterministic automotive networks. In *IEEE-SA Ethernet & IP @ Automotive Technology Day*, 2013.

[29] *CPLEX Optimization Studio*. IBM. [Online] http://www-01. ibm. com/software/integration /optimization/cplex-optimization-studio.

[30] Prachi Joshi, Sandeep K Shukla, Jean Pierre Talpin Inria, and Huafeng Yu. Mapping functional behavior onto architectural model in a model driven embedded system design. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 1624–1630. ACM, 2015.

[31] Prachi Joshi, SS Ravi, Soheil Samii, Unmesh D Bordoloi, Sandeep Shukla, and Haibo Zeng. Offset assignment to signals for improving frame packing in can-fd. In *Real-Time Systems Symposium (RTSS), 2017 IEEE*, pages 167–177. IEEE, 2017.

[32] Prachi Joshi, Haibo Zeng, Unmesh D Bordoloi, Soheil Samii, SS Ravi, and Sandeep K Shukla. The multi-domain frame packing problem for can-fd. In *Euromicro Conference on Real-Time Systems*, 2017.

[33] Minkoo Kang, Kiejin Park, and Myong-Kee Jeong. Frame packing for minimizing the bandwidth consumption of the FlexRay static segment. *IEEE Transactions on Industrial Electronics*, 60(9):4001–4008, 2013.

[34] Andreas Kern, Dominik Reinhard, Thilo Streichert, and Jürgen Teich. Gateway strategies for embedding of automotive can-frames into ethernet-packets and vice versa. In *International Conference on Architecture of Computing Systems*, pages 259–270, 2011.

[35] J. Kleinberg and E. Tardos. *Algorithm Design*. Pearson Education Inc., Boston, MA, 2006.

[36] Phil Koopman. A case study of toyota unintended acceleration and software safety. *Presentation. Sept*, 2014.

[37] Hermann Kopetz and Günther Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1):112–126, 2003.

[38] Hermann Kopetz, Astrit Ademaj, Petr Grillinger, and Klaus Steinhammer. The time-triggered ethernet (tte) design. In *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on*, pages 22–33. IEEE, 2005.

[39] S Kramer, D Ziegenbein, and A Hamann. Real world automotive benchmark for free. In *Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems (WATERS)*, 2015.

[40] Heungsoon Felix Lee and EC Sewell. The strip-packing problem for a boat manufacturing firm. *IIE transactions*, 31(7):639–651, 1999.

[41] Meng Liu, Moris Behnam, and Thomas Nolte. Schedulability analysis of gmf-modeled messages over controller area networks with mixed-queues. In *10th IEEE Workshop on Factory Communication Systems*, 2014.

[42] Weichen Liu, Mingxuan Yuan, Xiuqiang He, Zonghua Gu, and Xue Liu. Efficient sat-based mapping and scheduling of homogeneous synchronous dataflow graphs for throughput optimization. In *Real-Time Systems Symposium, 2008*, pages 492–504. IEEE, 2008.

[43] Andrea Lodi, Silvano Martello, and Michele Monaci. Two-dimensional packing problems: A survey. *European journal of operational research*, 141(2):241–252, 2002.

[44] Martin Lukasiewycz, Michael Glaß, Jürgen Teich, and Paul Milbredt. FlexRay schedule optimization of the static segment. In *Proceedings of the 7th IEEE/ACM international*

conference on Hardware/software codesign and system synthesis (CODES), pages 363–372. IEEE-ACM, 2009.

[45] Inês Lynce and Joao P Marques-Silva. On computing minimum unsatisfiable cores. In *Proceedings of the International Symposium on Theory and Applications of Satisfiability Testing*, pages 305–310, 2004.

[46] Frank D Murgolo. An efficient approximation scheme for variable-sized bin packing. *SIAM Journal on Computing*, 16(1):149–161, 1987.

[47] Marco Di Natale, Celso Luiz Mendes da Silva, and Max Mauro Dias Santos. On the applicability of an MILP solution for signal packing in CAN-FD. In *Proceedings of the IEEE 14th International Conference on Industrial Informatics (IEEE-INDIN)*, 2016.

[48] Nicolas Navet and Françoise Simonot-Lion. In-vehicle communication networks-a historical perspective and review. Technical report, University of Luxembourg, 2013.

[49] Roman Obermaisser. *Time-triggered communication*. CRC Press, 2011.

[50] José Fernando Oliveira, Alvaro Neuenfeldt Júnior, Elsa Silva, and Maria Antónia Carravilla. A survey on heuristics for the two-dimensional rectangular strip packing problem. *Pesquisa Operacional*, 36(2):197–226, 2016.

[51] Heikki Orsila. Optimizing algorithms for task graph mapping on multiprocessor system on chip. *Tampereen teknillinen yliopisto. Julkaisu-Tampere University of Technology. Publication*, 972, 2011.

[52] J. Palencia and M. Harbour. Schedulability analysis for tasks with static and dynamic offsets. In *IEEE Real-Time Systems Symposium*, 1998.

[53] Florian Polzlbauer, Iain Bate, and Eugen Brenner. Optimized frame packing for embedded systems. *IEEE Embedded Systems Letters*, 4(3):65–68, 2012.

[54] Paul Pop, Petru Eles, and Zebo Peng. Schedulability-driven communication synthesis for time triggered embedded systems. *Real-Time Systems*, 26(3):297–325, 2004.

[55] Paul Pop, Petru Eles, and Zebo Peng. Schedulability-driven frame packing for multicluster distributed embedded systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 4(1):112–140, 2005.

[56] Traian Pop, Petru Eles, and Zebo Peng. Schedulability analysis for distributed heterogeneous time/event triggered real-time systems. In *Real-Time Systems, 2003. Proceedings. 15th Euromicro Conference on*, pages 257–266. IEEE, 2003.

[57] Traian Pop, Paul Pop, Petru Eles, Zebo Peng, and Alexandru Andrei. Timing analysis of the flexray communication protocol. *Real-time systems*, 39(1-3):205–235, 2008.

[58] Rishi Saket and Nicolas Navet. Frame packing algorithms for automotive applications. *Journal of Embedded Computing*, 2(1):93–102, 2006.

[59] Kristian Sandstrom, C Norstom, and Magnus Ahlmark. Frame packing in real-time communication. In *Proceedings of the Seventh International Conference on Real-Time Computing Systems and Applications*, pages 399–403. IEEE, 2000.

[60] Alberto Sangiovanni-Vincentelli and Marco Di Natale. Embedded system design for automotive applications. *Computer*, 40(10), 2007.

[61] Alberto Sangiovanni-Vincentelli and Grant Martin. Platform-based design and software design methodology for embedded systems. *IEEE Design & Test of Computers*, 18(6): 23–33, 2001.

[62] Harmel Karam Singh and Abdou Youssef. Mapping and scheduling heterogeneous task graphs using genetic algorithms. Master's thesis, George Washington University, 1995.

[63] Till Steinbach, Franz Korf, and Thomas C Schmidt. Comparing time-triggered ethernet with flexray: An evaluation of competing approaches to real-time for in-vehicle networks. In *Factory Communication Systems (WFCS), 2010 8th IEEE International Workshop on*, pages 199–202. IEEE, 2010.

[64] Wilfried Steiner. An evaluation of smt-based schedule synthesis for time-triggered multi-hop networks. In *Real-Time Systems Symposium (RTSS), 2010 IEEE 31st*, pages 375–384. IEEE, 2010.

[65] Wilfried Steiner. Synthesis of static communication schedules for mixed-criticality systems. In *Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2011 14th IEEE International Symposium on*, pages 11–18. IEEE, 2011.

[66] Domitian Tamas-Selicean, Paul Pop, and Wilfried Steiner. Synthesis of communication schedules for ttethernet-based mixed-criticality systems. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 473–482. ACM, 2012.

[67] Bogdan Tanasa, Unmesh Dutta Bordoloi, Petru Eles, and Zebo Peng. Reliability-aware frame packing for the static segment of FlexRay. In *Proceedings of the Ninth ACM international conference on Embedded software*, pages 175–184. ACM, 2011.

[68] Daniel Thiele, Johannes Schlatow, Philip Axer, and Rolf Ernst. Formal timing analysis of can-to-ethernet gateway strategies in automotive networks. *Real-time systems*, 52(1): 88–112, 2016.

[69] KW Tindell, Hans Hansson, and Andy J Wellings. Analysing real-time communications: controller area network (CAN). In *Proceedings of Real-Time Systems Symposium*, pages 259–263. IEEE, 1994.

[70] AG TTTech Computertechnik. Ttethernet: Deterministic ethernet network-tttech, 2014.

[71] Gökhan Urul. *A Frame Packing Method to Improve the Schedulability of CAN and CAN-FD*. PhD thesis, Middle East Technical University, Turkey, 2015.

[72] Peter JM Van Laarhoven, Emile HL Aarts, and Jan Karel Lenstra. Job shop scheduling by simulated annealing. *Operations research*, 40(1):113–125, 1992.

[73] V. V. Vazirani. *Approximation Algorithms*. Springer, New York, NY, 2001.

[74] Vector. *DBC Communication Database for CAN*. Vector. [Online] https://vector.com/vi_candb_en.html.

[75] Ying Yi, Wei Han, Xin Zhao, Ahmet T Erdogan, and Tughrul Arslan. An ilp formulation for task mapping and scheduling on multi-core architectures. In *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09.*, pages 33–38. IEEE, 2009.

[76] Haibo Zeng and Marco Di Natale. An efficient formulation of the real-time feasibility region for design optimization. *IEEE Transactions on Computers*, 62(4):644–661, 2013.

[77] Haibo Zeng, Marco Di Natale, Arkadeb Ghosal, and Alberto Sangiovanni-Vincentelli. Schedule optimization of time-triggered systems communicating over the FlexRay static segment. *IEEE Transactions on Industrial Informatics*, 7(1):1–17, 2011.

[78] Haibo Zeng, Marco Di Natale, Arkadeb Ghosal, and Alberto Sangiovanni-Vincentelli. Schedule optimization of time-triggered systems communicating over the flexray static segment. *IEEE Transactions on Industrial Informatics*, 7(1):1–17, 2011.

[79] Haibo Zeng, Prachi Joshi, Daniel Thiele, Jonas Diemer, Philip Axer, Rolf Ernst, and Petru Eles. Networked real-time embedded systems. *Handbook of Hardware/Software Codesign*, pages 753–792, 2017.

[80] LuXi Zhao, Huagang Xiong, Zhong Zheng, and Qiao Li. Improving worst-case latency analysis for rate-constrained traffic in the time-triggered ethernet network. *IEEE Communications Letters*, 18(11):1927–1930, 2014.

[81] Wei Zheng, Qi Zhu, Marco Di Natale, and Alberto Sangiovanni Vincentelli. Definition of task allocation and priority assignment in hard real-time distributed systems. In *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, pages 161–170. IEEE, 2007.

[82] Husheng Zhou and Cong Liu. Task mapping in heterogeneous embedded systems for fast completion time. In *Proceedings of the 14th International Conference on Embedded Software*, page 22. ACM, 2014.