



CHALMERS
UNIVERSITY OF TECHNOLOGY



Multi-Layered In-Vehicle Communication Architectures

Mathematical Modeling and Optimization of Cost and Delay

Master's Thesis in Complex Adaptive Systems

Ali El Hage

Mattias Qian

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2026

www.chalmers.se

MASTER'S THESIS REPORT, 2026

Multi-Layered In-Vehicle Communication

Modeling and Optimization of Communication Architectures
Considering Cost and Delay

Ali El Hage, Mattias Qian



Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026

Multi-Layered In-Vehicle Communication
Modeling and Optimization of Communication Architectures
Considering Cost and Delay
Ali El Hage, Mattias Qian

© ALI EL HAGE, MATTIAS QIAN, 2026.

Supervisor: Malin Rau, Assistant Professor, Department of Mathematical Sciences
Examiner: Ann-Brith Strömberg, Professor, Department of Mathematical Sciences

Master's thesis report 2026
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden

Typeset in L^AT_EX
Gothenburg, Sweden 2026

Abstract

Modern bus platforms contain an increasing number of electronic control units (ECUs) distributed over long physical distances. This creates challenges for the in-vehicle communication architecture, since long cable routes, increasing traffic demands, and stricter timing requirements may reduce communication robustness and increase end-to-end delay. Controller Area Network (CAN) is widely used in vehicle systems due to its robustness, simplicity, and established industrial use. However, in long vehicle platforms, CAN-based communication may approach practical limitations related to cable length, signal integrity, and transmission delay.

This thesis investigates how a multi-layered in-vehicle communication architecture can be modeled and optimized by combining different communication technologies in a common bus architecture. The considered technologies include CAN, Ethernet, and AUTBUS. The objective is to determine where different technologies should be used and where converters should be placed in order to balance total system cost and communication delay.

A mixed-integer linear optimization model is developed to assign communication technologies to discretized bus segments and to determine the required gateway placements. The model includes cable costs, gateway costs, gateway delays, transmission delays, propagation delays, ECU interface constraints, and a maximum consecutive CAN length constraint. Since cost and delay are conflicting objectives, an epsilon-constraint method is used to generate Pareto-optimal architecture alternatives. A secondary fine-tuning model is also introduced to refine boundary positions after the discrete topology has been selected.

In addition to the mathematical model, physical gateway latency measurements are performed using a CANcase, CANalyzer, and CAPL scripts. The measurements show that gateway latency is not constant, but varies depending on transmission interval and network traffic. The measured latency is therefore used as a representative modelling parameter rather than an exact deterministic value.

The results show that optimization framework can identify meaningful trade-offs between cost and delay in a long bus communication architecture. The Pareto front provides several feasible design alternatives, ranging from low-cost solutions with higher delay to more expensive architectures with lower delay. The fine-tuning step further reduces cost for selected Pareto points, especially in the higher-cost region of the frontier. Overall, the thesis demonstrates that mixed-technology communication architectures can provide a structured approach for improving timing performance in long vehicle platforms, provided that gateway behavior, protocol constraints, and implementation complexity are carefully considered.

Keywords: In-vehicle communication, CAN, Ethernet, AUTBUS, gateway latency, bus architecture, mixed-integer linear programming, multi-objective optimization, Pareto frontier.

Acknowledgements

We would like to express our sincere gratitude to everyone who has supported and guided us throughout this master's thesis.

First, we would like to thank our academic supervisor, Malin Rau, at Chalmers University of Technology for her guidance, feedback, and support during the development of the mathematical modeling and optimization parts of the thesis. Her input has been important in improving the academic quality and structure of the work.

We would also like to thank our examiner, Ann-Brith Strömberg, at Chalmers University of Technology for her valuable feedback and for providing academic supervision and direction throughout the thesis process.

Furthermore, we would like to thank our industrial supervisor, Miguel Parreiras, at Volvo Buses for his continuous guidance, technical support, and valuable feedback during the project. His knowledge and insight on in-vehicle communication systems have been highly valuable for both the practical and theoretical parts of this work.

We would also like to extend our gratitude to our manager, Mateus Izar, at Volvo Buses for giving us the opportunity to carry out this thesis within Volvo Buses and for providing support throughout the project. His encouragement and practical perspective has been greatly appreciated.

Finally, we would like to thank Volvo Buses and Chalmers University of Technology for providing the resources, environment, and support needed to carry out this master's thesis.

List of Acronyms

The following list presents the acronyms used throughout this thesis, in alphabetical order.

AUTBUS	Automotive Universal Transport Bus
CAN	Controller Area Network
CAN FD	Controller Area Network Flexible Data-rate
CAN H	CAN High
CAN L	CAN Low
CANalyzer	Controller Area Network combined with analyzer
CAPL	Communication Access Programming Language
ECU	Electronic Control Unit
E/E	Electrical/Electronic
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FTTI	Fault Tolerant Time Interval
HARA	Hazard Analysis and Risk Assessment
ISO	International Organization for Standardization
LIN	Local Interconnect Network
SUMS	Software Update Management System
UN	United Nations
E2E	End-to-end
SEK	Swedish krona
s	second(s)
ms	millisecond(s)
μ s	microsecond(s)
h	hour(s)
FT	Fine Tuning
CANcase	Vector hardware interface for connecting a computer to a CAN network

Contents

List of Acronyms	ix
List of Figures	xiii
List of Tables	xvii
1 Introduction	1
1.1 Standard regulations for Volvo Buses	1
1.2 Purpose	2
1.3 Goals	3
1.4 Limitations	3
1.5 Method	4
1.5.1 The Weighted Sum Method	4
1.5.2 The ε -constraint method	4
2 Technical background	7
2.1 Communication challenges in long vehicle platforms	7
2.1.1 Bit period and voltage waveform	8
2.1.2 Signal sampling and bit detection	9
2.2 Networks	11
2.2.1 CAN-Network	11
2.2.2 Ethernet-Network	11
2.2.3 AUTBUS	12
2.3 Twisted cables	13
2.4 Gateway	14
2.5 Communication Delay	14
2.5.1 Propagation Delay	14
2.5.2 Transmission Delay	15
2.5.3 Gateway Delay	15
2.5.4 ECU Processing Delay	16
3 Mathematical modeling and solution methods	17
3.1 Mathematical Notation	17

3.2	Discretization	19
3.2.1	Boundary Classification	20
3.2.2	ECU Boundary Constraints	21
3.3	Two Optimization Objectives	23
3.3.1	Cost objective	23
3.3.2	Delay objective	24
3.4	Fine-tuning model	25
3.4.1	Fixed quantities	26
3.4.2	Objectives	26
3.4.3	Constraints	27
3.5	Mathematical properties and solution algorithms	28
3.5.1	Linear and mixed-integer linear programs	28
3.5.2	Dominated and non-dominated solutions	29
3.5.3	The Pareto frontier	30
3.5.4	The simplex method	30
3.5.5	The Branch-and-Bound	31
4	Test and Verification	35
4.1	Physical test	35
4.2	Communication Access Programming Language	36
4.3	Test results	37
4.3.1	Single-signal results	37
4.3.2	Multiple signals results	40
4.4	Gateway latency in mathematical model	44
5	Numerical Experiments and Results	47
5.1	Computational setup and data	47
5.2	Numerical Experiments	48
5.2.1	Test 1: Varying the Step Size	49
5.2.2	Test 2: Sweeping ε	50
6	Discussion	55
6.1	Test measurement discussion	55
6.2	Mathematical model discussion	57
6.3	Limitations of the measurement settings and models developed	59
7	Conclusions	61
	Bibliography	63

List of Figures

2.1	Comparison of CAN bit detection for a short and a long cable. In the short-cable case, the signal is stable at the sampling instant, while in the long-cable case the signal may still be transitioning or distorted, leading to uncertain bit detection.	10
2.2	Illustration of a typical CAN network: (a) physical bus wiring, where each blue CAN node represents an ECU containing a host, CAN controller, and CAN transceiver, with termination at both ends; and (b) differential voltage waveforms on CANH and CANL.	13
3.1	Example of a discretized bus topology with boundaries, gateways, and ECUs. Green dots represent regular interior boundaries (B^{reg}) where gateways may be placed, and red squares represent ECU boundaries (B^{ecu}) where additional protocol constraints apply.	21
3.2	Two scenarios for connecting an ECU to the main network. On the left, a bus-based protocol. On the right, a point-to-point protocol. . .	22
3.3	Illustration of the feasible and infeasible regions of the Pareto frontier. Red dot represents dominating solutions and the line represent the Pareto frontier.	30
3.4	Schematic illustration of a branch-and-bound tree for a minimization problem. Each node corresponds to a partial assignment of the binary variables, with branches created by fixing a fractional variable to either 0 or 1. The LP relaxation at each node provides a lower bound on the objective value attainable within that subtree. The incumbent value, obtained from the best feasible integer solution found so far, provides an upper bound on the optimal objective value. A node is pruned when its lower bound is greater than or equal to the incumbent upper bound, since it cannot contain a better integer solution. Feasible integer solutions (green) may update the incumbent value (red) and are then fathomed.	32
4.1	System setup with two gateways connected via AUTBUS, with CAN-case channels used for measurement. Generated with the TikZ package.	36

4.2	A photo of the final setup according to the design illustrated in Figure 4.1.	36
4.3	Gateway latency measured over 120 s for messages transmitted at intervals of 10 ms, corresponding to approximately 12,000 measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at 100 μ s. Consecutive measurements are connected by line segments.	37
4.4	Gateway latency measured over 120 s for messages transmitted at intervals of 100 ms, corresponding to approximately 1,200 measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at 100 μ s. Consecutive measurements are connected by line segments.	38
4.5	Gateway latency measured over 120 s for messages transmitted at intervals of 1000 ms, corresponding to approximately 120 measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at 100 μ s. Consecutive measurements are connected by line segments.	39
4.6	Gateway latency measured for the high-priority message during a 120 s multi-signal test. The high-priority message was transmitted at intervals of 1000 ms, corresponding to approximately 120 latency measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at 200 μ s. Consecutive measurements are connected by line segments.	40
4.7	Gateway latency measured for the low-priority message during a 120 s multi-signal test. The low-priority message was transmitted at intervals of 1000 ms, corresponding to approximately 120 latency measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at 200 μ s. Consecutive measurements are connected by line segments.	41
4.8	Gateway latency measured for the high-priority message during a 120 s multi-signal test. The high-priority message was transmitted at intervals of 100 ms, corresponding to approximately 1,200 latency measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at 200 μ s. Consecutive measurements are connected by line segments.	41
4.9	Gateway latency measured for the low-priority message during a 120 s multi-signal test. The low-priority message was transmitted at intervals of 100 ms, corresponding to approximately 1,200 latency measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at 200 μ s. Consecutive measurements are connected by line segments.	42

4.10	Gateway latency measured for the high-priority message during a 120 s multi-signal test. The high-priority message was transmitted at intervals of 10 ms, corresponding to approximately 12,000 latency measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at 200 μ s. Consecutive measurements are connected by line segments.	42
4.11	Gateway latency measured for the low-priority message during a 120 s multi-signal test. The low-priority message was transmitted at intervals of 10 ms, corresponding to approximately 12,000 latency measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at 200 μ s. Consecutive measurements are connected by line segments.	43
5.1	Pareto frontier with 100 ε -values.	51
5.2	Pareto frontier with 200 ε -values	51
5.3	Pareto-frontier structure observed in the denser ε -sweeps.	52
5.4	Selected ILP solutions with their corresponding fine-tuned solutions overlaid in orange.	54

List of Tables

2.1	Transmission delay per frame for each communication technology. . .	16
4.1	Measured gateway latency statistics in microseconds for the three test cases, i.e., the three transmission intervals of the single-signal test setup.	39
4.2	Measured gateway latency statistics in microseconds for high- and low-priority messages under multi-signal load conditions.	43
4.3	Gateway delay matrix for conversion between different communication protocols in microseconds.	44
5.1	Cable cost for CAN, 100BASE-T1, 1000BASE-T1, and AUTBUS. . .	47
5.2	Gateway costs for conversion between the communication protocols. .	48
5.3	Fine-tuning impact across the Pareto frontier. All costs are given in SEK and delays in milliseconds.	48
5.4	Effect of step size on model size, solution quality, and solve time for the cost-minimisation problem. Step size is given in meters, cost is given in SEK, and time is given in seconds unless otherwise stated. .	49
5.5	Selected ILP solutions and their corresponding fine-tuned solutions. Delay is given in milliseconds, and all costs are given in SEK. . . .	53

1

Introduction

In the early stages of the modern automotive industry, vehicle electrical systems were relatively simple. Cars contained only a small number of Electronic Control Units (ECUs), mainly responsible for basic functions such as braking assistance, steering support, and exterior lighting. As vehicle functionality expanded over time, the number of ECU's increased significantly. Modern vehicles, depending on their complexity, may contain more than 100 ECU's, each responsible for dedicated control, monitoring, or diagnostic tasks.

With this growing number of electronic components, using point-to-point wiring between all pairs of ECU's becomes impractical. Such an approach would result in excessive wiring complexity, increased weight and higher costs. To address these challenges, the automotive industry introduced network-based communication system's, allowing multiple ECU's to share common communication lines.

One of the most important of these systems is the Controller Area Network (CAN) network. CAN is a communication protocol that enables multiple ECU's to communicate over a shared two-wired logic bus. Instead of direct ECU-to-ECU connections, messages are broadcasted on the network, and each ECU processes only the messages relevant to its function.

1.1 Standard regulations for Volvo Buses

Modern buses developed by Volvo Buses must comply with established international standards and United Nations (UN) vehicle regulations to ensure safety, reliability, and continued approval throughout the vehicle lifecycle. These requirements directly influence how the electrical/electronic (E/E) architecture is designed, including the selection of communication technologies (e.g., CAN or Ethernet), network topology, gateway placement, and constraints related to timing and signal integrity. In this context, ISO 26262 and UN Regulation No.156 are particularly relevant, as they impose structured processes and technical expectations that must be considered when developing and modifying in-vehicle communication architectures.

ISO 26262 is the international functional safety standard that governs how E/E systems in road vehicles are developed so that faults do not lead to hazardous situations. The standard requires a safety lifecycle that includes hazard analysis and risk assessment (HARA), definition of safety goals, and implementation of safety

mechanisms that detect faults and transition the system into a defined safe state when necessary. A key concept is the Fault Tolerant Time Interval (FTTI), which specifies that faults must be detected and handled within a limited time before they can cause harm [1]. Depending on the safety concept and risk level, this is achieved through fail-safe behavior (controlled degradation or shutdown into a safe state) or fail-operational behavior (maintaining essential functionality through redundancy). For Volvo Buses, these requirements are directly linked to communication-dependent functions, since cable length, signal integrity, and network timing limitations in CAN or Ethernet can cause intermittent or systematic communication faults that must be detected and managed in a controlled manner.

UN Regulation No. 156 establishes requirements for a Software Update Management System (SUMS) to ensure that vehicle software updates are performed in a controlled and traceable way. The regulation requires manufacturers to assess and document the impact of updates prior to deployment, including effects on vehicle functions, network communication, and system performance (e.g., bus load, timing, routing, and gateway behavior). This is relevant for Volvo Buses because software updates may alter signal sets, message timing, or gateway routing, which can invalidate prior architecture assumptions if not managed correctly. Consequently, updates must not cause the in-vehicle networks (CAN/Ethernet) to exceed intended design limits, and the vehicle must remain verified and compliant after changes. While UN R156 primarily focuses on update management rather than functional safety [2], it complements ISO 26262 by reducing the risk that software modifications introduce unintended timing or communication degradations that could compromise reliability or safety.

1.2 Purpose

The purpose of this thesis is to investigate how communication delay and timing in long buses can be improved through architectural changes in the in-vehicle network. Modern bus platforms contain an increasing number of ECUs distributed over a large physical distance, which increases cable lengths and may lead to higher propagation delay, reduced signal margins, and timing-related communication issues, particularly in CAN-based domains. The work therefore focuses on evaluating and proposing a feasible network design that mitigates delay-related limitations while remaining compatible with practical packaging constraints and existing E/E architecture principles.

To achieve this, the bus communication network is modeled and analyzed to identify where replacing selected CAN segments with Ethernet, combined with appropriate gateway placement, can provide measurable timing benefits. The study quantifies expected performance improvements (e.g., reduced delay along critical paths) and balances them against design trade-offs such as added components, cost, and architectural complexity. The final outcome is a justified design recommendation and a reproducible method for selecting an efficient mix of CAN and Ethernet segments for long vehicle platforms.

1.3 Goals

The overall goal of this thesis is to deliver a technically justified and practically feasible recommendation for improving communication timing robustness in long bus platforms. Specifically, the thesis aims to produce (i) a clear characterisation of how cable length influences delay and signal integrity in the current architecture, (ii) a set of quantified constraints and design guidelines derived from the analysed limitations and applicable requirements, and (iii) an evaluated architectural proposal that mitigates delay-related issues while remaining implementable within realistic E/E-architecture conditions.

In addition, the thesis aims to provide a reproducible methodology that can be reused for similar vehicle platforms. This includes a structured modelling approach for the in-vehicle network, a consistent way to estimate and compare delay contributions across candidate segments provide net benefit without introducing unnecessary architectural complexity. The final deliverables are intended to support early-stage architecture decisions and to serve as a basis for further verification and implementation work within Volvo Buses.

1.4 Limitations

Since Volvo Buses supplies the chassis while the vehicle body is manufactured by an external bodybuilder, this project does not consider the complete electrical architecture of the final assembled bus. The scope is therefore limited to the chassis-level communication architecture and the information available for chassis-integrated systems. Electrical systems introduced by the bodybuilder are excluded from the analysis.

Furthermore, Volvo Buses offers multiple bus models, including different vehicle lengths, axle configurations, and so on. In this project, the analysis and calculations are performed for one selected bus variant. Although the numerical evaluation is specific to this configuration, the proposed architectural principles and optimization framework are intended to be generalizable to other bus models with similar structural constraints.

Another key limitation is that existing components cannot be redesigned or replaced. Modifying current ECU's to introduce native Ethernet-to-CAN capability would require hardware redesign, software redevelopment, revalidation according to functional safety standards, and coordination with suppliers. Therefore, the proposed solution must operate within the constraints of existing hardware. The introduction of entirely new hardware platforms is outside the scope of this study.

1.5 Method

The work is carried out in several stages. First, the in-vehicle communication architecture is represented as a discretized network and formulated as a mixed-integer linear optimization model. The model determines which communication technology should be used along different parts of the network and where gateways are required, while accounting for relevant architectural and physical constraints.

Second, physical measurements are performed to estimate gateway latency for different communication settings. These measurements are used to obtain representative parameter values for the optimization model, since suitable gateway-delay values were not available beforehand.

Third, the network architectures are evaluated with respect to two conflicting objectives: total system cost and end-to-end communication delay. Rather than selecting a single solution, several trade-off solutions are generated in order to identify Pareto-optimal compromises between the two objectives. The mathematical formulation and the methods used to obtain these solutions are described in detail in Chapter 3.

Finally, selected solutions are further refined using a secondary fine-tuning model. This model keeps the selected network topology fixed while allowing boundary positions to be adjusted continuously in order to improve the solution without violating the original design constraints.

1.5.1 The Weighted Sum Method

Several techniques exist for generating the Pareto frontier [3]. A common approach is the *weighted-sum method*, in which the two objectives are combined into a single weighted expression $\lambda C_{\text{tot}} + (1 - \lambda)D_{\text{tot}}$ and the weight λ is varied to produce different solutions.

The weighted-sum method is simple, but has a drawback. In the optimization model considered in this thesis, all decision variables are binary, meaning that the problem is a 0–1 integer linear program (ILP). The feasible set is therefore discrete, and the corresponding Pareto frontier consists of distinct objective-value points. For such a discrete and generally non-convex objective space, the weighted-sum method may fail to identify some Pareto-optimal solutions, regardless of how the weight λ is chosen [4].

1.5.2 The ε -constraint method

In this approach, one objective is optimized while the other is treated as an inequality constraint with a tunable upper bound ε . The cost is minimized as the primary objective, and the delay is constrained to remain below ε :

$$\begin{aligned}
& \min_{w, g, h} && C_{\text{tot}} \\
& \text{s.t.} && D_{\text{tot}} \leq \varepsilon \\
& && \text{constraints (3.1)–(3.12)} \\
& && w, g, h \in \{0, 1\}
\end{aligned} \tag{1.1}$$

The parameter ε are obtained from the two single-objective extreme solutions. Minimizing delay without any cost restriction yields the smallest achievable delay $D_{\min}$, while minimizing cost without any delay restriction yields the largest delay $D_{\max}$ associated with the cheapest feasible configuration. By sweeping ε across the interval $[D_{\min}, D_{\max}]$ and solving the constrained problem for each value, a sequence of non-dominated solutions is obtained, together approximating the Pareto frontier of the problem [3]. The resolution of this approximation is controlled by the number of values used, and a denser sampling produces a frontier at the cost of additional time.

In practice, the sweep may occasionally produce two solutions with identical cost and delay, or solutions that are marginally dominated due to solver tolerances. A post-processing filter is therefore applied to remove any dominated points before the frontier is presented.

2

Technical background

This chapter presents the technical background needed to understand the communication challenges studied in this thesis. It introduces the relevant in-vehicle communication technologies, physical cable effects, sampling principles, gateways, and delay of the components that form the basis for the mathematical optimization model formulated in Chapter 3.

2.1 Communication challenges in long vehicle platforms

In-vehicle communication networks are strongly influenced by the physical size and topology of the vehicle. In a long bus platform, electronic control units are distributed over a large distance, which means that the communication cables become significantly longer than in smaller vehicles. As the number of ECUs and vehicle functions increases, the communication network must also handle more signals, higher data traffic, and stricter timing requirements.

A cable has resistance, capacitance, and inductance distributed along its length. As the cable becomes longer, these electrical properties increase and affect the transmitted signal. The result can be slower voltage transitions, impairment, and reflections. These effects reduce the signal quality and make it more difficult for the receiver at the other end to correctly interpret the transmitted bits.

This is especially important in time-critical communication systems. The receiver must sample the signal at a specific point within each bit period. If the signal has not reached a clear voltage level before this sampling instant, the receiver may interpret the bit incorrectly. This can lead to bit errors, increased delay, and reduced communication reliability. Therefore, long cables do not only introduce physical signal degradation, but can also affect timing behavior and system robustness [5].

For bus vehicle platforms, these challenges are particularly relevant because long cable routes can push conventional in-vehicle communication technologies closer to their practical limits. CAN is robust, simple, and widely used in automotive systems. However, increasing cable length can increase signal propagation delay, reduce signal margins, and restrict the maximum achievable bit rate. In addition, high CAN network utilization, meaning that a large proportion of the available transmission capacity is occupied by message traffic, can cause messages to wait longer for access

to the shared communication channel due to CAN arbitration. The combination of long cable routes, high network utilization, and strict timing requirements can therefore make reliable and timely communication more difficult to maintain.

One possible way to address these challenges is to replace selected parts of the communication network with alternative or complementary technologies, such as Ethernet or AUTBUS. These technologies can provide higher data rates or improved suitability for long-distance communication, but they also introduce new design trade-offs related to cost, gateway placement, and implementation complexity. This motivates the need to study mixed-technology communication architectures rather than evaluating each technology in isolation.

The purpose of this technical background is therefore to explain the communication technologies and physical effects that are relevant for the thesis. Understanding why long cables create signal and timing problems is necessary before formulating the optimization model used to evaluate different network architectures.

2.1.1 Bit period and voltage waveform

Before introducing signal sampling, it is important to distinguish between the bit period and the physical voltage waveform on the cable. A bit is the smallest unit of digital information and can represent either a logical 0 (dominant) or a logical 1 (recessive). In a CAN network, each bit is assigned a fixed time interval during which it is transmitted on the bus. This time interval is called the bit period.

The bit period is determined by the configured CAN bit rate. For example, if the bit rate is 500 kbit/s, the duration of one bit is

$$T_{\text{bit}} = \frac{1}{\text{bitrate}} = \frac{1}{500 \times 10^3} \text{s} = 2 \mu\text{s}. \quad (2.1)$$

This means that every transmitted bit has a time window of 2 μs . This time window does not become longer when the cable becomes longer. It is fixed by the communication settings.

However, the voltage waveform within this bit period is affected by the physical cable. The voltage waveform describes how the electrical voltage on the bus changes over time when a bit is transmitted. Ideally, the voltage would change immediately from one level to another. In reality, this does not happen instantaneously. The cable has distributed resistance, capacitance, and inductance, which slow down and distort the voltage transition.

As the cable becomes longer, the total resistance, capacitance, and inductance increase. This can make the voltage transition slower and may also introduce reflections, ringing, or overshoot. Therefore, the bit period remains the same, but the signal shape inside the bit period changes. In other words, the receiver still expects to read the bit within the same fixed time window, but the voltage may not have

reached a clear and stable level when the decision is made.

This distinction is important because communication errors are not caused by the bit period changing. Instead, they occur because the physical voltage waveform may no longer settle fast enough within the available bit period.

2.1.2 Signal sampling and bit detection

After establishing the distinction between the fixed bit period and the physical voltage waveform, it is necessary to understand how the receiver converts the analog bus voltage into a digital bit value. In a CAN network, the receiver does not interpret the transmitted signal continuously. Instead, each bit is evaluated at a specific point in time within the bit period, known as the sampling instant. At this instant, the receiver compares the bus voltage with a decision threshold in order to determine whether the bit represents a logical 1 or 0, as explained in Subsection 2.1.1.

The bit period is determined only by the configured bit rate and is given in the formula (2.1).

The sampling instant is placed at a fraction $\alpha \in (0, 1)$ of the bit period. Choosing α relatively close to 1 places the sampling instant later in the bit period, allowing more time for the signal to propagate and settle before it is sampled

$$t_{\text{sample}} = \alpha T_{\text{bit}}. \quad (2.2)$$

A problem occurs when the physical voltage on the bus has not reached a sufficiently clear level before the sampling instant. This can happen when the cable becomes too long; as described in Subsection 2.1.1, longer cables can slow down and distort the voltage waveform. The consequence for sampling is that the signal may not have reached a sufficiently clear level at the sampling instant.

For reliable bit detection, the voltage at the sampling instant must exceed the receiver threshold, i.e.,

$$V(t_{\text{sample}}) > V_{\text{threshold}}. \quad (2.3)$$

If this condition is not fulfilled, the receiver samples the signal while it is still transitioning. In that case, the voltage may be close to the decision threshold, making it difficult for the receiver to determine whether the sampled bit is dominant or recessive. This can result in a bit error. In CAN communication, a frame is a structured sequence of bits transmitted as one complete communication unit. When a CAN node detects an error in an ongoing frame, it transmits an error frame, which is a predefined bit sequence used to notify the other nodes that an error has occurred. The ongoing transmission is then interrupted, and the affected CAN message is normally retransmitted. This increases communication delay and reduces the available network capacity. To illustrate the sampling problem, Figure 2.1 compares a stable signal that can be correctly detected with a distorted signal that may result in uncertain bit detection. If this condition is not fulfilled, the receiver samples the signal

while it is still transitioning. In that case, the voltage may be close to the decision threshold, making it difficult for the receiver to determine whether the sampled bit is dominant or recessive. This can result in a bit error. In CAN communication, a frame is a structured sequence of bits transmitted as one complete communication unit. When a CAN node detects an error in an ongoing frame, it transmits an error frame, which is a predefined bit sequence used to notify the other nodes that an error has occurred. The ongoing transmission is then interrupted, and the affected CAN message is normally retransmitted. This increases communication delay and reduces the available network capacity. To illustrate the sampling problem, Figure 2.1 compares a stable signal that can be correctly detected with a distorted signal that may result in uncertain bit detection.

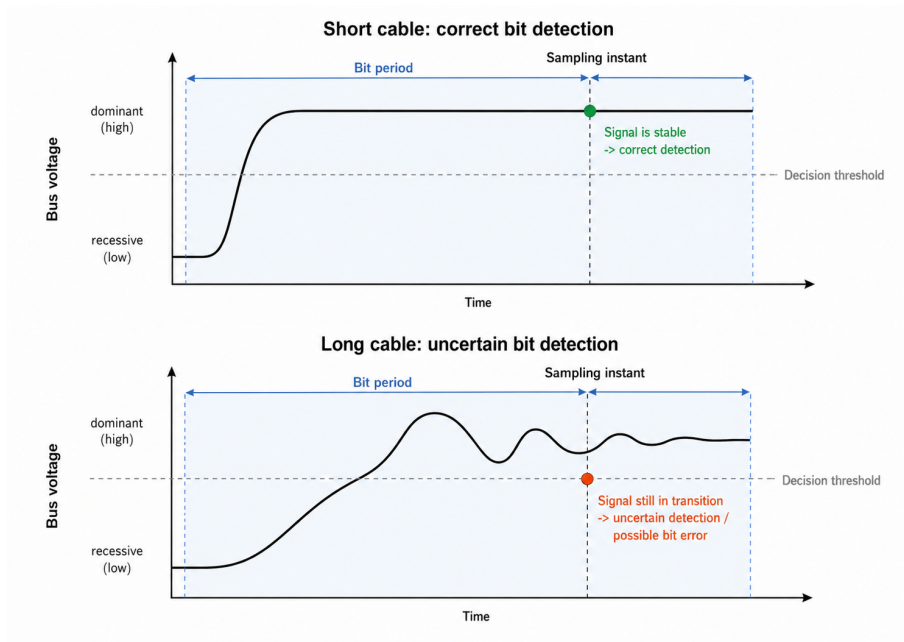


Figure 2.1: Comparison of CAN bit detection for a short and a long cable. In the short-cable case, the signal is stable at the sampling instant, while in the long-cable case the signal may still be transitioning or distorted, leading to uncertain bit detection.

Long cables can also cause signal reflections. Reflections occur when the signal encounters impedance changes, for example at cable ends, connectors, stubs, or incorrectly terminated sections. These reflected waves travel back along the cable and are superimposed on the original signal. As a result, the measured voltage may contain ringing, double steps, or oscillations. If any of these disturbances are present near the sampling instant, the receiver may interpret the bit incorrectly.

Therefore, long CAN cables create two main problems. First, they slow down the voltage transition, which reduces the time margin before sampling. Second, they increase the risk of reflections and signal distortion, which can disturb the voltage level at the exact moment when the receiver makes its decision. This is why cable length, topology, termination, and sampling point placement are important factors

when designing reliable CAN communication networks.

2.2 Networks

A modern bus contains several types of in-vehicle communication networks that interconnect ECU's and subsystems throughout the vehicle. Examples of such networks include CAN, Controller Area Network with Flexible Data-Rate (CAN FD), Local Interconnect Network (LIN), and, in newer architectures, automotive Ethernet. Each network technology has different characteristics and is therefore used for different functions depending on requirements such as data rate, robustness, cost, and timing.

However, not all communication technologies used in bus vehicles are relevant to the scope of this thesis. The following subsections therefore focus on the three technologies considered in this work: CAN, Automotive Ethernet, and AUTBUS.

2.2.1 CAN-Network

CAN is one of the most established in-vehicle communication protocols and is widely used for communication between electronic control units. It is based on a shared bus topology, where several nodes are connected to the same physical communication medium. Instead of sending data to a specific receiver, CAN messages are broadcast on the bus, and each ECU decides whether a received message is relevant based on the message identifier.

A key feature of CAN is its arbitration mechanism, where message priority is determined by the CAN identifier. This allows high-priority messages to access the bus before lower-priority messages when several nodes attempt to transmit at the same time. However, since all nodes share the same communication medium, the available bandwidth is limited and the timing behavior becomes more sensitive as bus load, cable length, and the number of connected nodes increase.

2.2.2 Ethernet-Network

Ethernet communication is governed by the same underlying physics as CAN. Ethernet cables also exhibit resistance, capacitance, and inductance, meaning that signals can experience rounded edges, attenuation, and reflections. Consequently, the analog waveform in Ethernet is distorted in a similar way to CAN as cable length increases.

However, the key difference between CAN and Ethernet is not the cable properties themselves, but the communication architecture. Ethernet is based on point-to-point links with well-defined characteristic impedance and without passive stubs. This creates a controlled transmission line and reduces reflections by design. In addition, each Ethernet link is electrically isolated from the next, which prevents

signal degradation from accumulating across the entire system length.

The most important distinction is that Ethernet employs active signal regeneration. When an Ethernet receiver captures a signal, it first interprets the analog waveform digitally and decides the bit values. It then transmits a newly regenerated signal on the next link, meaning that prior edge rounding, attenuation, and reflections do not propagate further through the system.

Ethernet also uses advanced line coding and clock recovery. The receiver recovers the clock directly from the data stream and applies equalization to compensate for cable attenuation and slow rise times. In other words, a non-ideal waveform is expected, and the system is explicitly designed to correct for these impairments.

Gigabit Automotive Ethernet

1000BASE-T1 is an automotive Ethernet standard that provides a data rate of 1 Gbit/s over a single twisted pair cable. Compared to Fast Automotive Ethernet (100BASE-T1), it offers significantly higher bandwidth and is therefore suitable for applications with larger data requirements, such as high-speed network communication, sensors, cameras, or centralized vehicle architectures. However, the higher data rate also places stricter requirements on the physical layer, including cable quality, signal integrity, and electromagnetic compatibility.

Fast Automotive Ethernet

100BASE-T1 is an automotive Ethernet standard that provides a data rate of 100 Mbit/s over a single twisted pair cable. It offers a lower data rate than 1000BASE-T1, but is typically less demanding in terms of physical-layer requirements. This makes it suitable for applications where higher bandwidth than CAN is needed, but where 1 Gbit/s is not necessary. In this thesis, 100BASE-T1 is considered as a possible intermediate communication technology between traditional CAN-based communication and higher-speed Ethernet solutions.

2.2.3 AUTBUS

AUTBUS is a communication technology considered in this thesis as a possible alternative or complement to CAN and Ethernet. It is intended for robust communication over longer cable distances and can therefore be relevant in long vehicle platforms where traditional CAN communication approaches its limitations.

One advantage of AUTBUS compared to Ethernet is that it is not limited to point-to-point communication. Instead, AUTBUS supports a multi-drop network topology, where several nodes can be connected to the same communication medium. This can be beneficial in a bus network, since ECUs are distributed along the vehicle and a pure point-to-point architecture may require additional switches, gateways, or cabling.

According to the AUTBUS technology description, AUTBUS can reach up to 254 multi-drop data network nodes over a distance of up to 500 meters using a twisted-pair cable, while operating at a transmission speed of 100 Mbit/s [6]. This makes it interesting for long vehicle platforms where both distance and network topology are important design factors.

However, AUTBUS is still a relatively new technology and is not currently used in the passenger bus industry. In this work, AUTBUS is included as an alternative network option for segments where CAN may be limiting, while accounting for the additional cost and delay introduced by gateways between AUTBUS and other technologies.

2.3 Twisted cables

All the network technologies described in Section 2.2 use a bus topology with two twisted wires. In for example CAN, there are two wires called CAN High and CAN Low, illustrated in Figure 2.2a and Figure 2.2b. These two wires are twisted together to reduce EMI, external noise, and impedance mismatches.

The ECUs are connected to the bus in parallel, as illustrated in Figure 2.2a. Each node typically consists of three main components: a microcontroller, a CAN controller, and a CAN transceiver.

Furthermore, termination resistors are placed at each physical end of the CAN bus. These termination resistors typically have a resistance of 120 Ω and are used to reduce signal reflections on the wire.

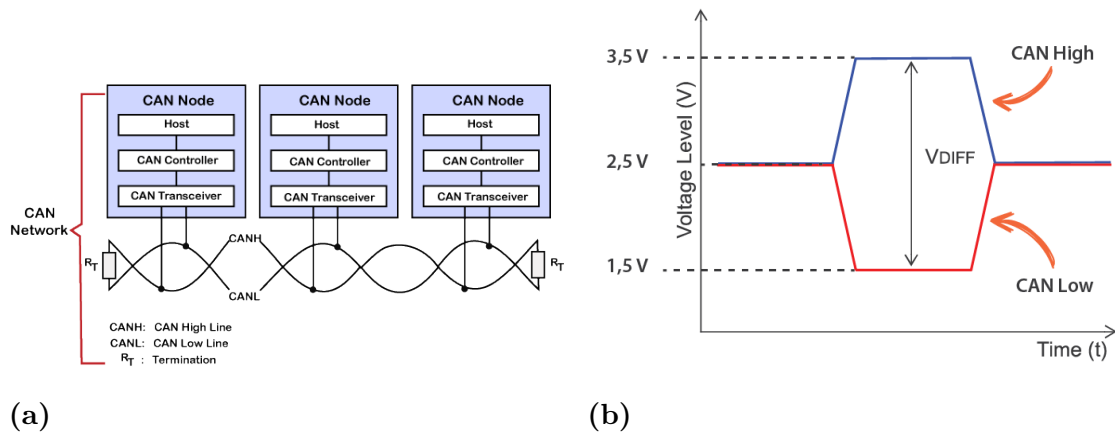


Figure 2.2: Illustration of a typical CAN network: (a) physical bus wiring, where each blue CAN node represents an ECU containing a host, CAN controller, and CAN transceiver, with termination at both ends; and (b) differential voltage waveforms on CANH and CANL.

Communication on the CAN bus is differential, meaning data is transmitted by measuring the voltage difference between CANH and CANL, rather than their absolute

voltages, which can be seen in Figure 2.2b. Using one wire with a higher voltage and the other with a lower voltage improves noise immunity, since external interference typically affects both wires equally and is therefore canceled out [7].

2.4 Gateway

A gateway, also called converter, is a device used to connect two different communication networks or protocols and enable data exchange between them. In automotive systems, gateways are commonly used when different parts of the vehicle rely on different communication technologies, such as CAN and Ethernet. Since these technologies operate with different data formats, speeds, and communication principles, a direct connection is not possible without a gateway.

The main purpose of a gateway is to receive messages from one network, process and convert them, and forward them to another network in a compatible format. In this way, gateways make it possible to integrate multiple communication technologies within the same vehicle system.

By using gateways, different communication technologies can coexist in one system. However, gateways also introduce additional cost, complexity, and delay, as discussed in Subsection 2.5.3. Therefore, any improvement in delay or scalability obtained by changing technology must be balanced against the additional gateway cost, gateway delay, and implementation complexity introduced by the mixed architecture.

2.5 Communication Delay

When a signal travels from one ECU to another through an in-vehicle communication network, it experiences several distinct sources of delay. Understanding each component is important for analysing the end-to-end timing behaviour of the network and for motivating the architectural choices made in this thesis. The total end-to-end delay can be decomposed into four main components: propagation delay, transmission delay, gateway delay, and ECU processing delay.

All of the following expressions are directly used in the mathematical model presented in Chapter 3, where they form the basis of the delay constraint in the multi-objective optimization problem.

2.5.1 Propagation Delay

Propagation delay is the time required for a signal to physically travel from one point to another along a cable. It depends on the propagation speed of the signal in the medium, which is a fraction of the speed of light $c \approx 3 \times 10^8$ m/s. For a cable of length l with propagation speed $v_p = r_p \cdot c$, where r_p is the velocity factor of the medium [8], the propagation delay is given by

$$d_p^{\text{prop}} = \frac{l}{v_p} = \frac{l}{r_p \cdot c}. \quad (2.4)$$

For twisted-pair cables used in CAN and AUTBUS, the velocity factor is typically around $r_p \approx 0.60$, while Ethernet cables exhibit propagation delays of approximately 5 ns/m, corresponding to a propagation velocity factor of about $r_p \approx 0.66\text{--}0.67$ [9, 10].

Propagation delay scales linearly with cable length and is generally small compared to the other delay components. For example, a 100 m CAN cable introduces a propagation delay of approximately $0.56\text{ }\mu\text{s}$, which is negligible compared to the transmission delay at 500 kbps.

2.5.2 Transmission Delay

Transmission delay, d^{tx} [s], is the time required to serialize a complete frame onto the communication medium. Unlike propagation delay, it is independent of cable length and depends on the frame size and bit rate of the communication technology, as

$$d^{\text{tx}} = \frac{f}{\beta}, \quad (2.5)$$

where f is the frame size in bits and β is the bitrate in bits per second. This distinction is important because propagation delay describes how fast the signal travels through the cable, while transmission delay describes how long it takes to put the entire frame onto the cable in the first place.

For CAN operating at 500 kbps with a worst-case extended frame of 131 bits, the transmission delay is given by

$$d_{\text{CAN}}^{\text{tx}} = \frac{131}{500 \times 10^3} \text{ s} \approx 262\text{ }\mu\text{s}. \quad (2.6)$$

For 100BASE-T1 Ethernet operating at 100 Mbps with a minimum frame of 672 bits, the transmission delay is given by

$$d_{100\text{BASE-T1}}^{\text{tx}} = \frac{672}{100 \times 10^6} \text{ s} \approx 6.7\text{ }\mu\text{s}. \quad (2.7)$$

This difference of approximately two orders of magnitude illustrates why transmission delay is the dominant delay term in CAN-based networks, and why replacing CAN segments with Ethernet can provide significant timing improvements. Table 2.1 summarises the transmission delay for each technology considered in this thesis.

2.5.3 Gateway Delay

Whenever data crosses a boundary between two different communication technologies, a gateway is required to translate and forward the data from one protocol

Table 2.1: Transmission delay per frame for each communication technology.

Technology	Bitrate (Mbps)	Frame size (bits)	d^{tx} (μs)
CAN	0.5	131	262.0
100BASE-T1	100	672	6.7
1000BASE-T1	1000	672	0.7
AUTBUS	100	131	1.31

to another. This process introduces an additional delay known as gateway delay. The gateway receives and processes the incoming frame, converts the data into the format required by the outgoing protocol, and transmits it on the outgoing link. The gateway delay depends on the specific pair of technologies involved and on the gateway implementation.

Two types of gateways are considered in this thesis. Regular gateways are installed at boundaries between adjacent intervals on the main network where the technology changes. ECU interface gateways are used when an ECU is connected to a part of the main network that is not CAN. Since the ECU communicates using CAN, a gateway is needed to convert between CAN and the technology used in the main network. Both regular gateways and ECU interface gateways add delay, and this delay increases each time the signal passes through a technology conversion along the end-to-end path.

2.5.4 ECU Processing Delay

ECU processing delay is the time spent inside the microcontroller at both the source and destination ECUs. It is distinct from transmission delay, which occurs on the bus, and instead reflects the latency accumulated within the software and hardware stack of the ECU itself.

At the source ECU, the processing delay includes the time for the application software to generate a signal value, the operating system to schedule the task, the protocol stack to format the data into a frame, and the driver to hand the frame to the CAN controller hardware. At the destination ECU, the processing delay includes the time for the CAN controller to raise an interrupt upon frame reception, the driver to read the frame from the hardware buffer, the protocol stack to parse it, and the application software to finally read the value.

Since the ECU processing delay occurs at both endpoints of every communication path, the total contribution is $2 \cdot d^{\text{ECU}}$. Importantly, this term is independent of the network architecture — it remains constant regardless of which technologies or gateways are selected along the path. It therefore does not influence the optimization but is included in the total delay expression for completeness and correctness.

3

Mathematical modeling and solution methods

This chapter presents the methodology used in this master's thesis, including the underlying reasoning behind the chosen approach and the development of the mathematical model used to analyze and optimize the communication architecture.

Problem description

Based on the technical background presented in Chapter 2, the main communication network is divided into intervals. Each interval is treated as a candidate segment that can be assigned one communication technology from a predefined set. The optimization model then determines which technology should be assigned to each interval and where gateways are required. Each feasible architecture is evaluated with respect to total system cost and communication delay, providing a systematic basis for identifying cost-efficient network designs that satisfy the model constraints.

3.1 Mathematical Notation

Before presenting the optimization model, the notation used in the mathematical formulation is summarized below, including indices, sets, parameters, and decision variables.

Sets

$I = \{1, \dots, n\}$	Set of intervals. Interval $i \in I$ is bounded by boundary $i - 1$ on its left and boundary i on its right
$B = \{0\} \cup I$	Set of boundaries.
$E \subseteq B$	Set of boundaries where ECUs are connected.
B^{int}	Set of interior boundaries.
B^{ecu}	Set of interior boundaries with ECU connections.
B^{reg}	Set of regular interior boundaries.

P	Set of candidate communication technologies.
P^{ECU}	Set of technologies allowed at interior ECU boundaries.
$P^{\text{ECU},\text{gw}}$	Set of ECU-compatible technologies requiring an interface gateway.

Indices

i	Index used for both intervals and boundaries, $i \in B$.
j	Index for starting interval in the CAN length constraint.
p	Index for communication technology.
p_L	Communication technology on the left side of a boundary.
p_R	Communication technology on the right side of a boundary.

Parameters

l_i	Length of interval i .
c_p^{cab}	Cable cost per meter for communication technology p .
$c_{p_L p_R}^{\text{gw}}$	Cost of a gateway converting from technology p_L to technology p_R .
$d_{p_L p_R}^{\text{gw}}$	Delay of a gateway converting from technology p_L to technology p_R .
d^{prop}	Propagation delay along the considered path.
d^{ECU}	ECU processing delay.
D_{max}	Maximum allowed end-to-end delay.
L_{CAN}	Maximum allowed consecutive CAN length.
k_j	Number of intervals from starting interval j needed to exceed L_{CAN} .
p^{ECU}	Communication protocol used by the ECU, assumed to be CAN.

Variables

w_{ip}	Binary variable equal to 1 if interval i is assigned communication technology p , and 0 otherwise. For an interior boundary i , $w_{i,p}$ denotes the technology of the interval to the left of the boundary and $w_{i+1,p}$ denotes the technology of the interval to the right.
$g_{ip_L p_R}$	Binary variable equal to 1 if a regular gateway is installed at boundary i between technology p_L and technology p_R , and 0 otherwise.

h_{ip}

Binary variable equal to 1 if an ECU-interface gateway is installed at boundary i for technology p , and 0 otherwise.

3.2 Discretization

The physical bus of total length L is discretized into a finite set of intervals $I = \{1, \dots, n\}$ which are separated by an ordered set of boundary points, indexed by the set $B = \{0\} \cup I$. The boundaries are created by merging the start and end positions 0 and L , all ECU positions, and any additional manually specified positions, followed by adding uniformly spaced step points with spacing Δs in meters. This produces an ordered set of $n + 1$ boundary positions and a corresponding set of n intervals, where interval $i \in I$ spans from boundary $i - 1$ to boundary i with length l_i , which is a fixed parameter determined by the discretization.

The step size Δs controls the resolution of the discretization. A smaller step size gives the optimizer more flexibility in placing gateways at precise positions along the bus. Each interval is treated as a candidate segment that can be assigned exactly one communication technology from the set

$$P = \{\text{CAN}, \text{100BASE-T1}, \text{1000BASE-T1}, \text{AUTBUS}\}.$$

To model the technology assignment, the binary variables $w_{ip} \in \{0, 1\}$ are introduced, which equals 1 if interval $i \in I$ is assigned technology $p \in P$, and 0 otherwise. Each interval must be assigned exactly one technology, modeled by the constraints.

$$\sum_{p \in P} w_{ip} = 1 \quad \forall i \in I. \quad (3.1)$$

At each boundary between two consecutive intervals, the model compares the technologies assigned to the adjacent intervals. If the technologies differ, a gateway is installed at that boundary to enable a protocol conversion between them. The binary variable $g_{i,p_L,p_R} \in \{0, 1\}$ equals 1 if a gateway converting from technology p_L to technology p_R is placed at regular boundary $i \in B^{\text{reg}}$, and 0 otherwise. The gateway cost for such a conversion is denoted $c_{p_L p_R}^{\text{gw}}$ and the corresponding processing delay is denoted $d_{p_L p_R}^{\text{gw}}$. The following constraints ensure that a gateway must be placed whenever both adjacent intervals match its protocol pair:

$$g_{i,p_L,p_R} \geq w_{i,p_L} + w_{i+1,p_R} - 1 \quad \forall i \in B^{\text{reg}}, \forall p_L, p_R \in P : p_L \neq p_R. \quad (3.2)$$

The following constraints ensure that a gateway can only be active if both adjacent intervals actually use the protocols it connects, preventing the model from placing a gateway where no protocol transition occurs, the constraints:

$$g_{i,p_L,p_R} \leq w_{i,p_L} \quad \forall i \in B^{\text{reg}}, \forall p_L, p_R \in P : p_L \neq p_R. \quad (3.3)$$

$$g_{i,p_L,p_R} \leq w_{i+1,p_R} \quad \forall i \in B^{\text{reg}}, \forall p_L, p_R \in P : p_L \neq p_R. \quad (3.4)$$

Together these constraints ensure that a gateway is placed if and only if a protocol transition occurs. Additionally, at most one gateway is permitted at any single boundary, preventing multiple simultaneous protocol conversions at the same point, which is modelled as

$$\sum_{p_L \in P} \sum_{p_R \in P \setminus \{p_L\}} g_{i,p_L,p_R} \leq 1 \quad \forall i \in B^{\text{reg}}. \quad (3.5)$$

While the gateway linking constraints ensure that protocol transitions are correctly modelled, an additional physical limitation must be considered for CAN. Due to signal integrity constraints arising from the electrical properties of long cables, as discussed in Chapter 2, no contiguous stretch of CAN cable may exceed a maximum length of $L^{\text{CAN}} = 30$ meters. To enforce this, for each starting interval $j \in I$, the parameter k_j is defined as the smallest number of consecutive intervals starting at j whose cumulative length exceeds L^{CAN} :

$$k_j := \min \left\{ k \in \{1, \dots, n - j + 1\} \left| \sum_{i=j}^{j+k-1} l_i > L^{\text{CAN}} \right. \right\} \quad \forall j \in I. \quad (3.6)$$

This parameter identifies the shortest window of consecutive intervals starting at j that would violate the CAN length limit if all assigned CAN. At least one interval within each such window must therefore be assigned a technology other than CAN, i.e.,

$$\sum_{i=j}^{j+k_j-1} (1 - w_{i,\text{CAN}}) \geq 1 \quad \forall j \in I. \quad (3.7)$$

If no such window exists starting at j (i.e., if $k_j = 1$), meaning the remaining bus length from j to the end is shorter than L^{CAN} , no constraint is needed for that starting point.

3.2.1 Boundary Classification

The boundary points are classified into two types, as illustrated in Figure 3.1. Interior boundaries that coincide with an ECU position form the set B^{ecu} , shown as red squares, where additional protocol constraints apply as discussed below. All remaining interior boundaries form the set B^{reg} , shown as green dots, where gateways may be freely placed whenever the protocol changes between adjacent intervals. Together these two sets form the set of all interior boundaries $B^{\text{int}} = B^{\text{reg}} \cup B^{\text{ecu}}$, excluding the two physical endpoints at positions 0 and L . The full set of all boundaries is

denoted $B = \{0\} \cup I$, and the set of all boundaries where an ECU is located is denoted $E \subseteq B$, which includes both interior ECU boundaries and any ECU located at the physical endpoints.

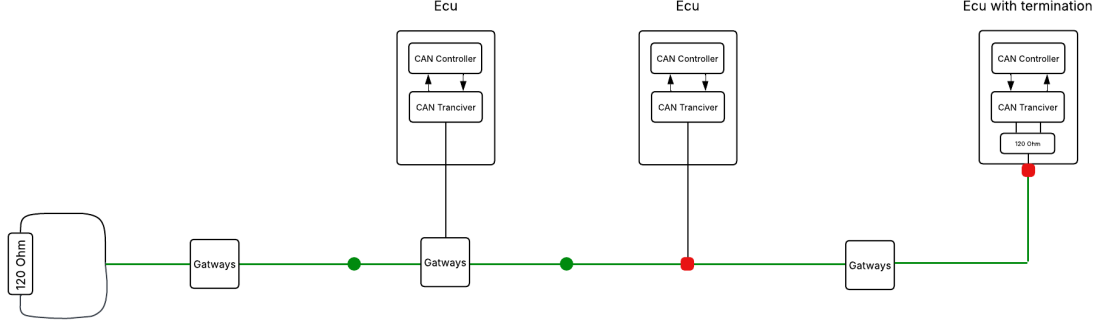


Figure 3.1: Example of a discretized bus topology with boundaries, gateways, and ECUs. Green dots represent regular interior boundaries (B^{reg}) where gateways may be placed, and red squares represent ECU boundaries (B^{ecu}) where additional protocol constraints apply.

The two physical endpoints form a special case. Unlike interior boundaries, no gateway can be placed outside the bus, meaning there is no interval to the left of boundary 0 or to the right of boundary n where a stub gateway could be inserted. If an ECU is present at either endpoint, the adjacent interval is therefore forced to use CAN so that the ECU can connect directly to the main network without a gateway, modelled as:

$$w_{1,\text{CAN}} = 1 \quad \text{if } 0 \in E. \quad (3.8)$$

$$w_{n,\text{CAN}} = 1 \quad \text{if } n \in E. \quad (3.9)$$

If no ECU is present at an endpoint, no protocol restriction applies and the adjacent interval may use any technology from P .

3.2.2 ECU Boundary Constraints

As illustrated in Figure 3.2, two scenarios are considered for connecting an ECU to the main network. In the bus topology case, shown on the left, the main network uses a bus-based protocol such as CAN or AUTBUS, which supports a multi-drop topology where multiple nodes can connect to the same physical medium.

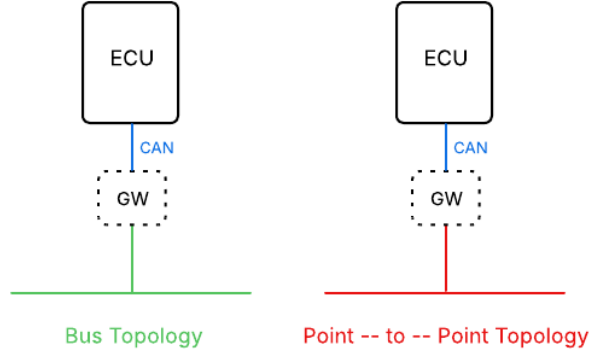


Figure 3.2: Two scenarios for connecting an ECU to the main network. On the left, a bus-based protocol. On the right, a point-to-point protocol.

The ECU connects to the main network via a stub gateway that converts between the main network protocol and the CAN protocol used by the ECU controller. In the point-to-point topology case, shown on the right, the main network uses Ethernet, which is a point-to-point protocol that does not support branching. Connecting an ECU stub to an Ethernet segment would require splitting the link, which is physically not possible without introducing an additional switch. Since adding switches is outside the scope of this model, Ethernet is excluded from ECU boundaries entirely. This leaves only CAN and AUTBUS as permitted technologies at ECU boundaries, forming the set $P^{\text{ECU}} = \{\text{CAN}, \text{AUTBUS}\}$, modelled as

$$w_{i,p} = 0, \quad w_{i+1,p} = 0 \quad \forall i \in B^{\text{ecu}}, \forall p \in P \setminus P^{\text{ECU}}. \quad (3.10)$$

Furthermore, since no three-port gateway exists that can simultaneously bridge two different backbone protocols and connect to the ECU, the technology must be identical on both sides of an ECU boundary. Having different technologies on each side would require a gateway with three connections, one for the left interval, one for the right interval, and one for the ECU stub, which does not exist in practice. These limitations are modelled as

$$w_{i,p} = w_{i+1,p} \quad \forall i \in B^{\text{ecu}}, \forall p \in P. \quad (3.11)$$

It should be noted that in the model, the stub gateway is associated with the ECU boundary point rather than a physical position on the stub itself. This is a modelling simplification, but the logic remains consistent. When the adjacent intervals use AUTBUS, the stub gateway is activated to convert from AUTBUS to CAN for the ECU interface. When the adjacent intervals use CAN, no stub gateway is needed since the ECU can connect directly to the main network. The set of ECU-compatible technologies requiring a stub gateway is denoted $P^{\text{ECU,gw}} =$

$P^{\text{ECU}} \setminus \{\text{CAN}\} = \{\text{AUTBUS}\}$. The binary variable $h_{i,p} \in \{0, 1\}$ equals 1 if a stub gateway converting from technology p to CAN is placed at ECU boundary $i \in B^{\text{ecu}}$, and 0 otherwise. The stub gateway is activated if and only if the adjacent intervals use a non-CAN ECU-compatible technology, modelled as

$$h_{i,p} = w_{i,p} \quad \forall i \in B^{\text{ecu}}, \forall p \in P^{\text{ECU}, \text{gw}}. \quad (3.12)$$

3.3 Two Optimization Objectives

Two quantities are of practical interest when designing the communication architecture, the total cost of the network and the worst-case end-to-end signal delay between the ECUs. These two quantities are inherently in conflict, since reducing delay typically requires faster but more expensive technologies, while reducing cost favours slower, cheaper protocols such as CAN. The model is therefore formulated as a bi-objective optimization problem in which both quantities are expressed as linear functions of the decision variables introduced in the previous sections.

3.3.1 Cost objective

The total cost of a network configuration is composed of three contributions, the cost of the cable used, the cost of the gateways, and the cost of the stub gateways required at ECU boundaries.

The cable cost depends on the length of each interval and on the cable cost per meter c_p^{cab} associated with the technology p assigned to each interval. The total cost of the cable is obtained by summing the contribution of the intervals

$$C_{\text{cable}} = \sum_{i \in I} \sum_{p \in P} \ell_i c_p^{\text{cab}} w_{i,p} \quad (3.13)$$

The gateway cost at a regular boundary follows directly from the gateway variables g_{i,p_L,p_R} . Whenever a gateway is active between two adjacent intervals using different technologies, the corresponding cost c_{p_L,p_R}^{gw} is added to the total, modelled as

$$C_{\text{gw}}^{\text{reg}} = \sum_{i \in B^{\text{reg}}} \sum_{p_L \in P} \sum_{p_R \in P \setminus \{p_L\}} c_{p_L,p_R}^{\text{gw}} g_{i,p_L,p_R}. \quad (3.14)$$

A similar contribution arises at ECU boundaries. As described in Section 3.2.2, an ECU connected to a non-CAN main network requires a stub gateway to convert the protocol to the CAN interface. The activation of stub gateway is done by the variable $h_{i,p}$, and the cost is

$$C_{\text{gw}}^{\text{ecu}} = \sum_{i \in B^{\text{ecu}}} \sum_{p \in P^{\text{ECU}, \text{gw}}} c_{p, \text{CAN}}^{\text{gw}} h_{i,p}. \quad (3.15)$$

The total cost objective, which is to be minimized, the sum of the three contributions

$$C_{\text{tot}} = C_{\text{cable}} + C_{\text{gw}}^{\text{reg}} + C_{\text{gw}}^{\text{ecu}}. \quad (3.16)$$

3.3.2 Delay objective

The second objective is the worst-case end-to-end delay experienced by a signal traveling between the first and the last ECU on the bus. Four sources of delay are considered, propagation delay, transmission delay, gateway processing delay, and a fixed processing delay at the ECUs.

The four delay components together form the total end-to-end delay experienced by a signal traveling from a source ECU to a destination ECU through the in-vehicle network. Summing over all intervals and boundaries along the propagation path I^{path} , the total delay is expressed as

$$\begin{aligned} D = & \underbrace{\sum_{i \in I^{\text{path}}} \sum_{p \in P} d_p^{\text{prop}} l_i w_{ip}}_{\text{propagation delay}} + \underbrace{\sum_{i \in I^{\text{path}}} \sum_{p \in P} d_p^{\text{tx}} w_{ip}}_{\text{transmission delay}} + \underbrace{\sum_{i \in B^{\text{reg}}} \sum_{p_L \in P} \sum_{p_R \in P \setminus \{p_L\}} d_{p_L p_R}^{\text{gw}} g_{i p_L p_R}}_{\text{gateway delay}} \\ & + \underbrace{\sum_{i \in B^{\text{ecu}}} \sum_{p \in P^{\text{ECU}, \text{gw}}} d_{p^{\text{ECU}}}^{\text{gw}} \cdot h_{ip}}_{\text{ECU gateway delay}} + \underbrace{2 \cdot d^{\text{ECU}}}_{\text{processing delay}}. \end{aligned} \quad (3.17)$$

The propagation delay depends on the physical length of each interval and on the propagation delay per meter $d_p^{\text{prop}} = l/v_p^{\text{prop}}$ of the assigned technology, which was discussed in Subsection 2.5.1. Only the intervals located between the first and the last ECU contribute to this delay, and this subset is denoted $I^{\text{ecu}} \subseteq I$. The propagation delay is thus expressed as

$$D_{\text{prop}} = \sum_{i \in I^{\text{ecu}}} \sum_{p \in P} \ell_i d_p^{\text{prop}} w_{i,p}. \quad (3.18)$$

The transmission delay is the time required to clock one full worst-case frame onto the bus and is given by $d_p^{\text{tx}} = f/\beta$. As with the propagation delay, only intervals on the communication path between the two outermost ECUs are relevant, and the transmission delay is expressed as

$$D_{\text{tx}} = \sum_{i \in I^{\text{ecu}}} \sum_{p \in P} d_p^{\text{tx}} w_{i,p}. \quad (3.19)$$

Each gateway placed along the communication path adds its own processing delay. Both regular gateways and ECU stub gateways contribute, leading to the expression

$$D_{\text{gw}} = \sum_{i \in B^{\text{reg}}} \sum_{\substack{p_L, p_R \in P \\ p_L \neq p_R}} d_{p_L, p_R}^{\text{gw}} g_{i, p_L, p_R} + \sum_{i \in B^{\text{ecu}}} \sum_{p \in P^{\text{ECU}, \text{gw}}} d_{p, \text{CAN}}^{\text{gw}} h_{i, p}. \quad (3.20)$$

Finally, a constant processing delay d^{ecu} is added twice, once in the transmitting ECU and once in the receiving ECU. The total end-to-end delay, which is to be minimized, is therefore

$$D_{\text{tot}} = D_{\text{prop}} + D_{\text{tx}} + D_{\text{gw}} + 2d^{\text{ecu}}. \quad (3.21)$$

3.4 Fine-tuning model

The optimization problem formulated in Section 3.2 and Section 3.3 operates on a discretized version of the bus, where the candidate boundary positions are fixed in advance by the resolution given by the step size Δs . In practice, this means that, for a given choice in the ε -constraint model (Equation 1.5.2), the resulting protocol assignment and gateway placement constitute the optimal MILP solution for the chosen ε -point. However, the positions of the interval boundaries are restricted to this discrete step size and may therefore not be optimal in a continuous sense.

The resolution also affects the computational complexity of the model. Let $n = |I|$ denote the number of intervals and $m = |P|$ the number of candidate communication technologies. The main model contains $\mathcal{O}(nm^2)$ binary variables and $\mathcal{O}(nm^2)$ constraints, primarily due to the gateway variables and their associated linking constraints for every possible pair of technologies at each boundary. Since the number of technologies is fixed in this study ($m = 4$), both the number of variables and the number of constraints grow linearly with the number of intervals, i.e., as $\mathcal{O}(n)$. Nevertheless, because the problem is a MILP, its worst-case solution time is exponential in the number of binary variables. Consequently, decreasing Δs increases the number of intervals and can substantially increase the solver runtime.

To address this, a second optimization model is introduced as a post-processing step. Once a particular configuration has been selected from the Pareto frontier, the protocol assignment and the gateway locations are kept fixed, while the model is allowed to adjust the interval lengths continuously. This second stage does not change the structure of the network: the selected protocols, the number of gateways, and the type of each gateway remain identical to those of the chosen Pareto solution. Only the position of each boundary along the logical bus is refined. The fine-tuning model contains $\mathcal{O}(n)$ continuous variables and $\mathcal{O}(n)$ constraints and is therefore computationally simpler than the main mixed-integer model.

3.4.1 Fixed quantities

Since the protocol assignment and the gateway placements are inherited from the chosen Pareto solution, several quantities that depend on the binary variables in the first model become constants in the fine-tuning stage. These constants can be computed directly from the chosen Pareto solutions and replace the objective form the main model, as well as the delay expressions without modelling them as variables.

The fixed gateway cost contains the contribution of all gateways at regular boundaries and at ECU boundaries

$$C_{\text{gw}}^{\text{fix}} = \sum_{i \in B^{\text{reg}}} \sum_{p_L \in P} \sum_{p_R \in P \setminus \{p_L\}} c_{p_L, p_R}^{\text{gw}} g_{i p_L p_R}^* + \sum_{i \in B^{\text{ecu}}} \sum_{p \in P \setminus \{\text{CAN}\}} c_{p, \text{CAN}}^{\text{gw}} h_{ip}^*. \quad (3.22)$$

where p_i^* denotes the protocol assigned to interval i in the chosen Pareto solution. The fixed gateway delay is defined analogously

$$D_{\text{gw}}^{\text{fix}} = \sum_{\substack{i \in B^{\text{reg}} \\ p_i^* \neq p_{i+1}^*}} d_{p_i^*, p_{i+1}^*}^{\text{gw}} + \sum_{\substack{i \in B^{\text{ecu}} \\ p_i^* \neq \text{CAN}}} d_{p_i^*, \text{CAN}}^{\text{gw}}. \quad (3.23)$$

The transmission delay depends only on which protocol is assigned to each interval, and not on the length of the interval. It is therefore also a constant in the fine-tuning model, given by

$$D_{\text{tx}}^{\text{fix}} = \sum_{i \in I^{\text{ecu}}} d_{p_i^*}^{\text{tx}}. \quad (3.24)$$

The only decision variables in the fine-tuning model are the continuous interval lengths $\ell_i \geq \ell_{\min}$ for all $i \in I$, where $\ell_{\min}$ is a small positive lower bound that prevents intervals from collapsing to zero length. The protocol assignment p_i^* is no longer a decision but a fixed input.

3.4.2 Objectives

Two variants of the fine-tuning model are considered, corresponding to the two original objectives. In the cost-oriented variant, the model minimizes the total cable cost by aiming to shrink the intervals assigned to expensive protocols and expand those assigned to cheaper ones, with the objective to minimize

$$\min_{\ell_i, i \in I} \sum_{i \in I} \ell_i c_{p_i^*}^{\text{cab}}. \quad (3.25)$$

In the delay-oriented variant, the model minimizes the propagation delay along the communication path between the first and the last ECU by aiming to shrink the intervals assigned to slower protocols, with the objective to minimize

$$\min_{\ell_i, i \in I} \sum_{i \in I^{\text{ecu}}} \ell_i d_{p_i^*}^{\text{prop}}. \quad (3.26)$$

3.4.3 Constraints

To preserve the structure of the chosen topology, the intervals are not allowed to move freely along the entire bus. Instead, the bus is divided into a number of fixed *zones* delimited by anchor points that must remain in place: the two physical endpoints, every ECU position, and any manually specified position. The set of zones is denoted Z , and for every zone $z \in Z$ the set of intervals contained within it is denoted $I_z \subseteq I$ with total physical length L_z . The lengths of the intervals inside a zone may be redistributed freely, but their sum must equal the original length of that zone, modelled as

$$\sum_{i \in I_z} \ell_i = L_z \quad \forall z \in Z. \quad (3.27)$$

These constraints ensure that ECUs and other anchor points remain at their original physical positions, since the zones between them cannot expand or contract.

The maximum CAN length constraint introduced in (3.7) is also enforced in the fine-tuning model. For every contiguous stretch of intervals assigned to CAN in the chosen topology, the sum of the corresponding lengths must not exceed the physical signal integrity limit L_{CAN} , which is modelled as

$$\sum_{i \in S_{\text{CAN}}} \ell_i \leq L_{\text{CAN}} \quad \forall S_{\text{CAN}} \subseteq I. \quad (3.28)$$

where each S_{CAN} denotes a maximal set of consecutive intervals with $p_i^* = \text{CAN}$.

Finally, an upper bound equal to the delay of the chosen Pareto solution is imposed on the total delay. This directly prevents the fine-tuning step from yielding a higher delay, while the cost cannot increase either, since the original boundary configuration remains a feasible candidate under this constraint and therefore provides a lower bound on the achievable cost. Both objective values are thus guaranteed to be no larger than their respective value at the chosen Pareto point. Letting D^* denote the total delay of the chosen Pareto solution, the constraint reads

$$D_{\text{prop}}(\ell) + D_{\text{tx}}^{\text{fix}} + D_{\text{gw}}^{\text{fix}} + 2d^{\text{ecu}} \leq D^*. \quad (3.29)$$

where $D_{\text{prop}}(\ell) = \sum_{i \in I^{\text{ecu}}} \ell_i d_{p_i^*}^{\text{prop}}$ is the variable propagation delay. This constraint ensures that the fine-tuning model can only improve the chosen objectives.

3.5 Mathematical properties and solution algorithms

Both optimization models presented in this chapter are solved using Gurobi, a commercial mathematical programming solver widely used for large-scale linear, mixed-integer linear, and quadratic optimization problems [11]. Gurobi is interfaced through its Python API, `gurobipy`, which allows the models to be constructed programmatically and solved without leaving the Python environment. Although the same solver is used for both stages, the two models differ fundamentally in mathematical structure and computational complexity, and Gurobi therefore applies different algorithms to each. The fine-tuning model is a linear program (LP) containing only continuous variables and can be solved in polynomial time using interior-point methods. In contrast, the first model is a mixed-integer linear program (MILP), whose binary variables introduce combinatorial complexity and make the general problem NP-hard.

3.5.1 Linear and mixed-integer linear programs

A *Linear Program (LP)* in standard form is an optimization problem of the type

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{3.30}$$

where $x \in \mathbb{R}^n$ is the vector of decision variables, $c \in \mathbb{R}^n$ is the cost vector, $A \in \mathbb{R}^{m \times n}$ is the constraint matrix, and $b \in \mathbb{R}^m$ is the right-hand-side vector. The feasible region $\{x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0\}$ is a convex polyhedron, and the objective is a linear function of x . Linear programs are polynomial-time solvable in theory, as established by the ellipsoid and interior-point methods [12, 13]. However, commonly used implementations, particularly simplex-based algorithms, can exhibit exponential worst-case complexity. Despite this theoretical worst-case behaviour, they generally perform efficiently in practice.

A *Mixed-Integer Linear Program (MILP)* extends this formulation by requiring a subset of the variables to take integer values:

$$\begin{aligned} \min_{x, y} \quad & c^\top x + d^\top y, \\ \text{s.t.} \quad & Ax + By \leq b, \\ & x \geq 0, \quad y \in \{0, 1\}^q, \end{aligned} \tag{3.31}$$

where $x \in \mathbb{R}^n$ is the vector of continuous decision variables and $y \in \{0, 1\}^q$ is the vector of binary decision variables. The vectors $c \in \mathbb{R}^n$ and $d \in \mathbb{R}^q$ contain the objective coefficients associated with the continuous and binary variables, respectively.

Furthermore, $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m \times q}$ are the corresponding constraint matrices, while $b \in \mathbb{R}^m$ is the right-hand-side vector.

The integrality requirement on y makes the feasible set nonconvex and introduces the combinatorial difficulty of the problem. MILP is NP-hard [14], meaning that no algorithm is known that solves every MILP instance in polynomial time.

The first model presented fits the MILP form (3.31) with $n = 0$, q equal to the total number of binary variables $w_{i,p}$, g_{b,p_L,p_R} , and $h_{b,p}$, and the constraints (3.1)–(3.12). The fine-tuning model fits the LP form (3.30) with $x = \ell$ and the constraints described in Section 3.4

3.5.2 Dominated and non-dominated solutions

Consider two feasible network configuration x and y , with cost and delay values (C_x, D_x) and (C_y, D_y) , x is said to *dominate* y if it is at least as good as y in both objectives and strictly better in at least one of them, that is

$$C_x \leq C_y \quad \text{and} \quad D_x \leq D_y, \quad \text{with at least one strict inequality.} \quad (3.32)$$

If such a solution x exists, then y is called a *dominated* solution, since it can be improved in one objective without any sacrifice in the other and would therefore never be chosen by a rational decision maker. A solution for which no other feasible solution dominates it is called *non-dominated* or *Pareto optimal* [3]. For such a solution, any further improvement in one objective necessarily requires a worsening of the other.

It is worth emphasizing that a Pareto-optimal solution is not necessarily optimal with respect to either objective individually. One Pareto-optimal solution may have a lower cost but a higher delay than another. Neither solution dominates the other because improving one objective requires sacrificing the other. A feasible solution is Pareto optimal if no other feasible solution exists that is at least as good in both objectives and strictly better in at least one. During the solution process, a solution may be non-dominated among the solutions found so far. However, this does not by itself verify that the solution is Pareto optimal, since an undiscovered feasible solution may dominate it.

To illustrate this, consider three known feasible configurations A , B , and C with cost and delay values (5000, 2.0), (8000, 1.0), and (6000, 1.5), respectively, where cost is expressed in SEK and delay in ms. Configuration A has the lowest cost but the highest delay, while configuration B has the lowest delay but the highest cost. Configuration C is intermediate in both objectives. None of these three configurations dominates another. They are therefore non-dominated within the set of configurations found so far. However, this does not prove that they belong to the true Pareto frontier, since other feasible configurations may exist that dominate them.

For example, suppose that a fourth feasible configuration D is found with a cost of 4000 SEK and a delay of 1.0 ms. Configuration D dominates A and C because it is strictly better in both objectives. It also dominates B because it has the same delay and a strictly lower cost. Consequently, only D is non-dominated among the four known configurations.

3.5.3 The Pareto frontier

The set of all non-dominated solutions forms the Pareto frontier of the problem ;see[15]. Plotted in the cost–delay plane, it appears as a set of points that traces the best achievable trade-off between the two objectives: every point on the frontier corresponds to a network design where neither cost nor delay can be reduced without worsening the other.

Points above the Pareto frontier correspond to feasible but dominated solutions. Points below the frontier are infeasible, which is demonstrated in Figure 3.3

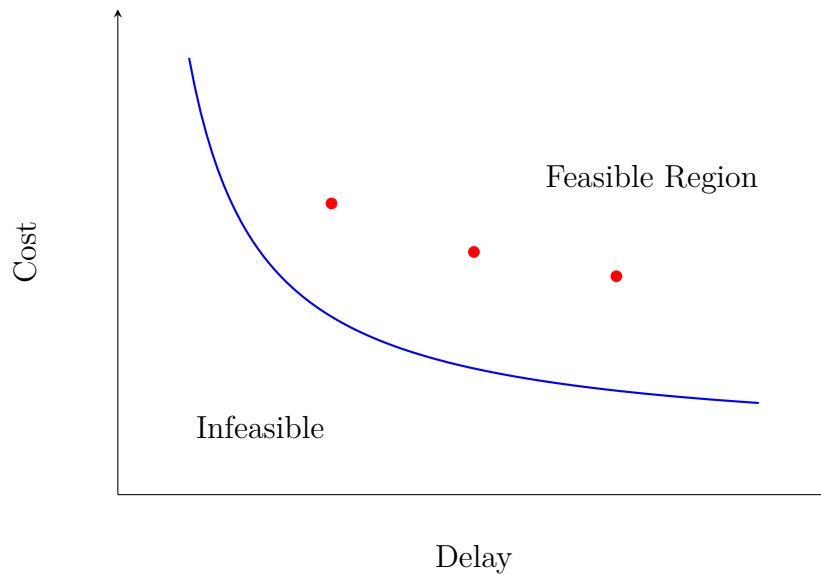


Figure 3.3: Illustration of the feasible and infeasible regions of the Pareto frontier. Red dot represents dominating solutions and the line represent the Pareto frontier.

The Pareto frontier therefore provides a complete map of the meaningful design alternatives and allows the final architecture to be selected based on the relative importance assigned to cost and delay.

3.5.4 The simplex method

The fine-tuning model differs fundamentally from the first one. Since the protocol assignment and the gateway placements are inherited from the chosen Pareto so-

lution, all binary variables are fixed to their optimal values and no longer appear as decisions. The interval lengths ℓ_i , which were fixed parameters in the MILP determined by the spatial discretization, are now promoted to continuous decision variables. All constraints remain linear in ℓ_i , so the fine-tuning model is a linear program of the form (3.30). Linear programs are dramatically less computationally demanding than mixed-integer programs. The feasible region

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0\} \quad (3.33)$$

is a convex polyhedron, and a fundamental result of linear programming theory states that, if an optimal solution exists, at least one optimal solution is attained at a vertex of $\mathcal{P}$ [16, 17]. The search can therefore be restricted to the finite set of vertices rather than the entire feasible region.

Gurobi solves linear programs using the *simplex method*. The method maintains a sequence of vertices of $\mathcal{P}$, called *basic feasible solutions*, and moves between them along edges of the polyhedron. At each iteration, the algorithm selects a non-basic variable whose increase would improve the objective and moves to the adjacent vertex obtained by allowing that variable to enter the basis. Letting $x^{(k)}$ denote the current vertex at iteration k and $x^{(k+1)}$ the next, the procedure satisfies

$$c^\top x^{(k+1)} < c^\top x^{(k)} \quad \forall k. \quad (3.34)$$

The algorithm terminates when no such improving direction exists, which by the optimality conditions of linear programming implies that the current vertex is optimal [18].

For the fine-tuning LP considered here, the dual simplex is typically preferred because the problem is derived from a MILP by fixing binary variables, the resulting LP inherits a dual-feasible basis from the relaxation, making the dual simplex the natural starting point [17].

In addition to the simplex method, Gurobi also implements an *interior-point* or *barrier* method, which approaches the optimum through the interior of the polyhedron rather than along its boundary [13, 19]. Interior-point methods have polynomial worst-case complexity and are often preferred for very large problems instance, but for the small fine-tuning LP considered in this thesis the simplex method is fast enough that the choice has little practical impact.

The fine-tuning model is solved once per chosen Pareto point. Since it contains only continuous variables and no integrality requirements, no branching is needed, and the entire problem is solved in a fraction of a second.

3.5.5 The Branch-and-Bound

The first model (described in Section 3.2 and Section 3.3,) is a bi-objective mixed-integer linear program, which can be expressed as an ϵ -constrained MILP as defined

in (1.1). A small instance with q binary variables has 2^q possible assignments, which becomes computationally intractable to enumerate even for moderate values of q .

To solve MILPs efficiently, Gurobi uses the *branch-and-bound* algorithm [20, 21], enhanced by a number of additional techniques that have become standard in modern solvers. The algorithm proceeds in three repeating phases, illustrated in Figure 3.4.

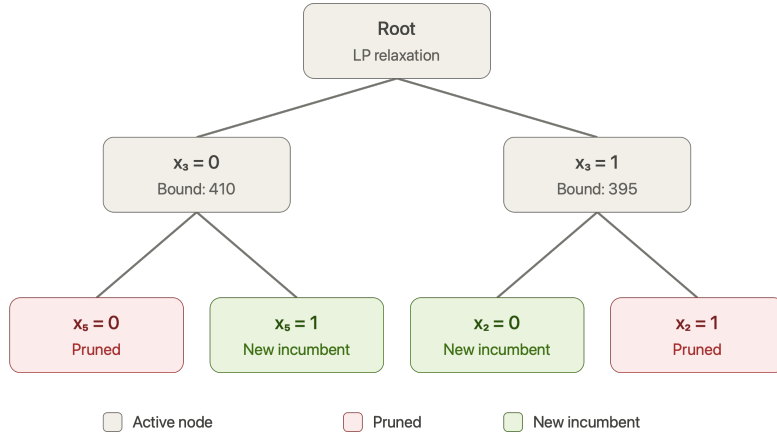


Figure 3.4: Schematic illustration of a branch-and-bound tree for a minimization problem. Each node corresponds to a partial assignment of the binary variables, with branches created by fixing a fractional variable to either 0 or 1. The LP relaxation at each node provides a lower bound on the objective value attainable within that subtree. The incumbent value, obtained from the best feasible integer solution found so far, provides an upper bound on the optimal objective value. A node is pruned when its lower bound is greater than or equal to the incumbent upper bound, since it cannot contain a better integer solution. Feasible integer solutions (green) may update the incumbent value (red) and are then fathomed.

The first step is to ignore the integrality requirement and treat all binary variables as continuous values in the interval $[0, 1]$. Given a MILP of the form (3.31), its *LP relaxation* is obtained by replacing $y \in \{0, 1\}^q$ with $y \in [0, 1]^q$:

$$\begin{aligned}
 \min_{x, y} \quad & c^\top x + d^\top y, \\
 \text{s.t.} \quad & Ax + By \leq b, \\
 & x \geq 0, \quad 0 \leq y \leq 1.
 \end{aligned} \tag{3.35}$$

The solution of (3.35) provides a lower bound on the optimal objective value of the original MILP, since the relaxed problem is less constrained and can therefore not have a worse optimal value. The relaxation is typically not a valid solution to the original MILP, because y may take fractional values that may have no physical interpretation, but it serves as a reference for the search.

If the relaxation produces a fractional value, the algorithm selects such a variable and splits the problem into two subproblems.

$$\text{LP}_{y_j=0} : \begin{cases} \min & c^\top x + d^\top y \\ & Ax + By \leq b \\ & x \geq 0 \\ & 0 \leq y \leq 1. \end{cases} \quad \text{LP}_{y_j=1} : \begin{cases} \min & c^\top x + d^\top y \\ & Ax + By \leq b \\ & x \geq 0 \\ & 0 \leq y \leq 1. \end{cases} \quad (3.36)$$

This produces a tree of subproblems, called the *branch-and-bound tree*, where each node corresponds to a partial assignment of the binary variables.

Let $z_{\text{node}}^{\text{LP}}$ denote the optimal objective value of the LP relaxation at a given node in the tree, and let z^{inc} denote the best valid integer solution found so far, called the *incumbent*. If

$$z_{\text{node}}^{\text{LP}} \geq z^{\text{inc}} \quad (3.37)$$

then no integer solution within that branch can improve on the incumbent, and the entire subtree can be discarded without further exploration. This pruning step is what prevents the search from growing in practice, since large portions of the tree are typically eliminated before being explored.

In addition to the basic branch-and-bound framework, Gurobi applies several techniques that accelerate the search considerably [11]. A *presolve* phase simplifies the model before the search begins by removing redundant constraints, fixing variable values that can be deduced from logical implications, and tightening variable bounds. During the search, *cutting planes* are added: these are linear inequalities $\alpha^\top y \leq \beta$ that are satisfied by every integer-feasible solution but violated by the current fractional LP optimum, thereby tightening the relaxation without removing any valid solution.

In this thesis, the first model is solved once for every value of ε in the epsilon-constraint sweep. Each solve returns the optimal protocol assignment and gateway placement for the corresponding delay bound, from which the cost and delay of that Pareto point are computed.

4

Test and Verification

In this chapter, the tests performed in this master's project are shown and explained. The underlying ideas and reasoning behind the chosen tests are also explained.

4.1 Physical test

For this work, two different gateway types were provided by Volvo Buses: an AUTBUS-to-Ethernet gateway and a CAN-to-AUTBUS gateway. A Vector CANcase was also provided. The CANcase is a hardware interface used to connect a computer running CANalyzer or CANoe to a CAN network, enabling CAN frames to be transmitted, received, and monitored. An initial objective was to design an experimental setup for estimating the latency introduced when a message is forwarded through a gateway. To enable this, a test wiring arrangement was prepared. Four wires were used in total, consisting of two CAN High (CAN H) conductors and two CAN Low (CAN L) conductors, which were twisted into pairs to form a CAN connection. One end of the wires was then soldered to a connector in order to interface the setup with the relevant hardware. The complete setup was intended to connect a CANcase and one ECU at one side side of the test arrangement, and another ECU at the opposite side.

The main idea of the experiment was the following. ECU A transmits a CAN signal, which is monitored at the transmitting side through one channel of the CANcase. The signal then enters the first gateway, where it is converted from CAN to AUTBUS. After propagating through the AUTBUS link, it enters a second gateway, where it is converted back from AUTBUS to CAN. Finally, the signal reaches ECU B, where a second channel of the CANcase is used to detect the signal at the receiving side.

By comparing the measured time at the transmitting and receiving sides, the total propagation time through the complete path can be determined. This measured time includes transmission through the cables as well as the delay introduced by the two gateways. In this way, the setup makes it possible to estimate the total end-to-end latency for a signal traveling from one side of the system to the other via two gateway conversions.

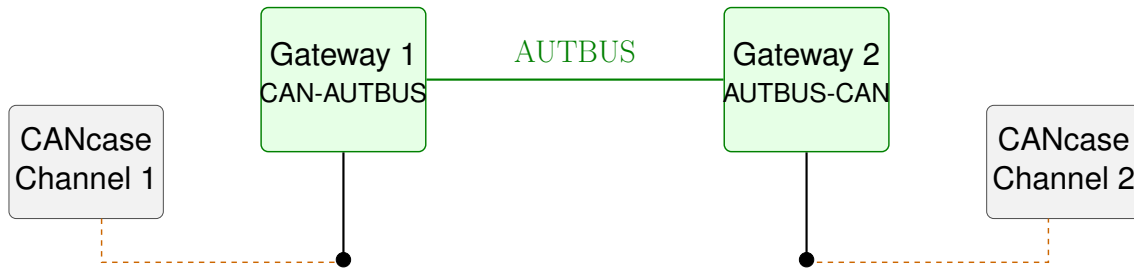


Figure 4.1: System setup with two gateways connected via AUTBUS, with CANcase channels used for measurement. Generated with the TikZ package.

The setup according to Figure 4.1 was set up successfully and connected to CANalyzer.

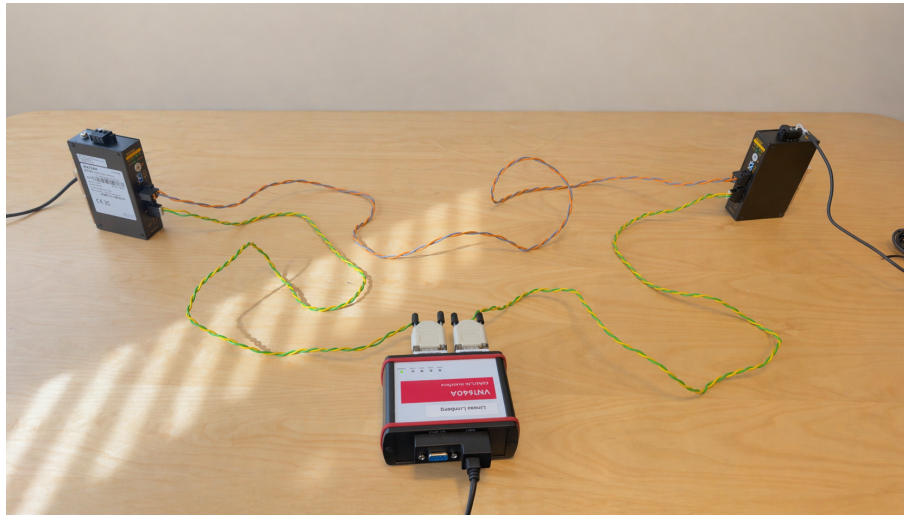


Figure 4.2: A photo of the final setup according to the design illustrated in Figure 4.1.

4.2 Communication Access Programming Language

Communication Access Programming Language (CAPL) was used in CANalyzer to calculate the latency measurements performed in the tests, the results of which are reported in Section 4.3. A CAPL script was written to detect the transmitted CAN message on one CANcase channel and the corresponding received message on the other channel. By comparing the timestamps of the two message events, the script calculated the gateway latency for each transmitted sample. The code also stored and printed the total number of samples together with the average, minimum, and maximum latency for each test case. This setup made it possible to evaluate the gateway behavior for different transmission intervals and traffic conditions in a repeatable way.

4.3 Test results

Subsection 4.3.1, presents the measured gateway latency over time for three different transmission intervals: 10 ms, 100 ms, and 1000 ms. In these tests, only one signal was transmitted and analyzed in order to study its latency behavior in isolation. The latency was recorded continuously during a measurement period of 120 seconds for each test case, allowing both the average delay level and its variation over time to be measured.

In Subsection 4.3.2, the same signal was analyzed again, but this time together with many additional simultaneously transmitted signals. The purpose of these tests was to investigate how the measured latency is affected when the network is exposed to additional traffic and possible interference. This makes it possible to compare how the gateway behavior changes with different message transmission frequencies, both in isolation and under more loaded communication conditions, and to identify possible outliers, fluctuations, or recurring patterns in the measured delay.

4.3.1 Single-signal results

For the 10 ms case, illustrated in Figure 4.3 the latency signal appears the most dense and irregular, with frequent fluctuations and several pronounced spikes. This suggests that the gateway delay is more variable when messages are transmitted at a higher rate. The overall level is also higher compared to the other cases, which is consistent with the highest average latency being observed in this test; see Table 4.1

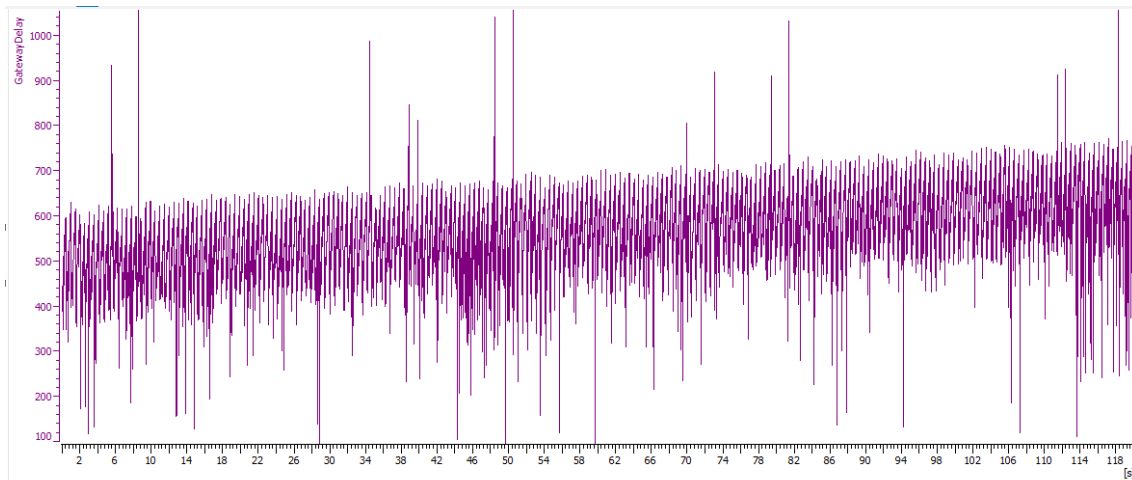


Figure 4.3: Gateway latency measured over 120 s for messages transmitted at intervals of 10 ms, corresponding to approximately 12,000 measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at 100 μ s. Consecutive measurements are connected by line segments.

In the 100 ms case, illustrated in Figure 4.4, approximately 1,200 latency measurements were collected during the 120-second test, compared with approximately 12,000 measurements in the 10 ms case. This smaller number of measurements

results in fewer overlapping line segments in the figure. The measured latency fluctuates over time, with a few isolated peaks representing temporarily higher latency values. Despite these peaks, this test produced the lowest average latency among the three transmission intervals.

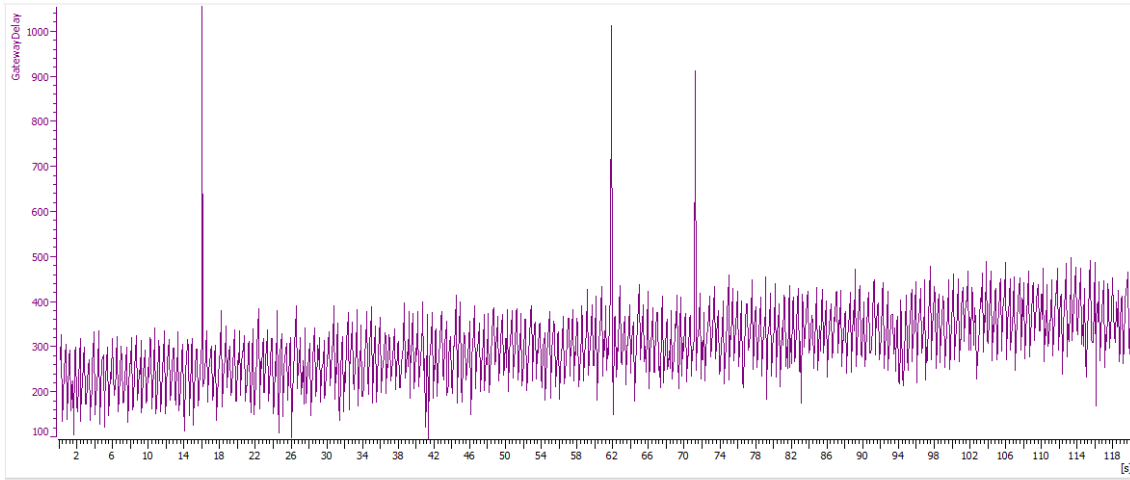


Figure 4.4: Gateway latency measured over 120 s for messages transmitted at intervals of 100 ms, corresponding to approximately 1,200 measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at 100 μ s. Consecutive measurements are connected by line segments.

For the 1000 ms case, illustrated in Figure 4.5, only approximately 120 latency measurements were collected during an equally long measurement period, compared with approximately 12,000 and 1,200 measurements in the 10 ms and 100 ms cases, respectively. Since consecutive measurements are separated by 1 s, their positions are more widely spaced along the horizontal axis. This makes the individual measurements and the connecting line segments easier to distinguish. However, the smoother visual appearance should not by itself be interpreted as evidence of lower latency variability. The average latency in this case is higher than for the 100 ms case but lower than for the 10 ms case.

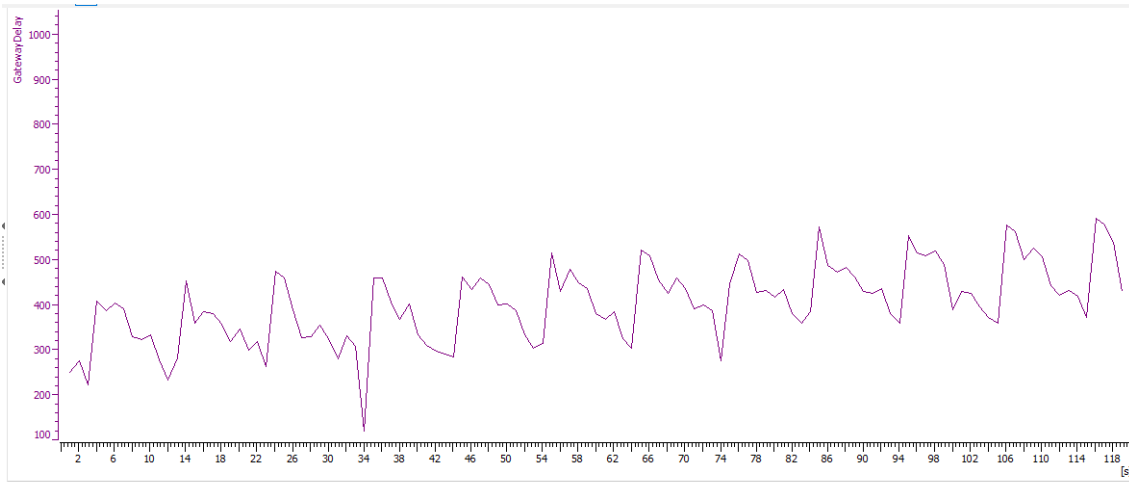


Figure 4.5: Gateway latency measured over 120 s for messages transmitted at intervals of 1000 ms, corresponding to approximately 120 measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at 100 μs . Consecutive measurements are connected by line segments.

CAPL in CANalyzer was programmed to output the total number of samples, as well as the average, minimum, and maximum latency for each separate test. The results are shown in Table 4.1.

Test case	Number of samples	Average latency	Minimum latency	Maximum latency
10 ms	11999	567.63	86.78	1088.49
100 ms	1199	304.34	93.06	1067.53
1000 ms	119	401.96	119.88	591.21

Table 4.1: Measured gateway latency statistics in microseconds for the three test cases, i.e., the three transmission intervals of the single-signal test setup.

Single-signal short discussion

The one-signal tests provide a baseline measurement of the gateway latency when no additional network traffic is present. Although only one message was transmitted and analyzed, the latency still varied over time, indicating that the measured delay is affected by more than only the physical transmission path.

The 10 ms case showed the highest average latency and the most irregular behavior, while the 100 ms case resulted in the lowest average latency. The 1000 ms case contained fewer samples and appeared smoother in the plot. These results are used as a reference for comparison with the multi-signal tests presented in Subsection 4.3.2.

The median latency would have provided a more robust measure for comparing the three test cases, since the arithmetic mean can be strongly affected by isolated large latency values.

4.3.2 Multiple signals results

Figure 4.6 and Figure 4.7 show the measured gateway latency when the analyzed signal is transmitted with a 1000 ms interval while additional signals are present on the network. The high-priority signal shows a relatively stable and smooth latency behavior, with only moderate variations over time. In contrast, the low-priority signal shows larger disturbances and several clear latency drops, especially in the later part of the measurement series. This indicates that lower-priority messages are more affected when several signals are transmitted simultaneously.

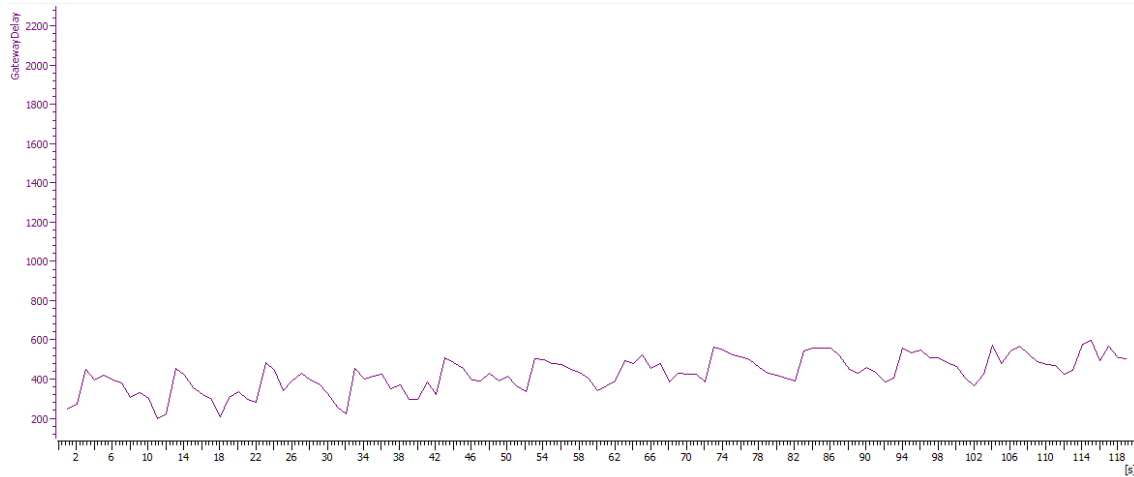


Figure 4.6: Gateway latency measured for the high-priority message during a 120 s multi-signal test. The high-priority message was transmitted at intervals of 1000 ms, corresponding to approximately 120 latency measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at $200\ \mu\text{s}$. Consecutive measurements are connected by line segments.

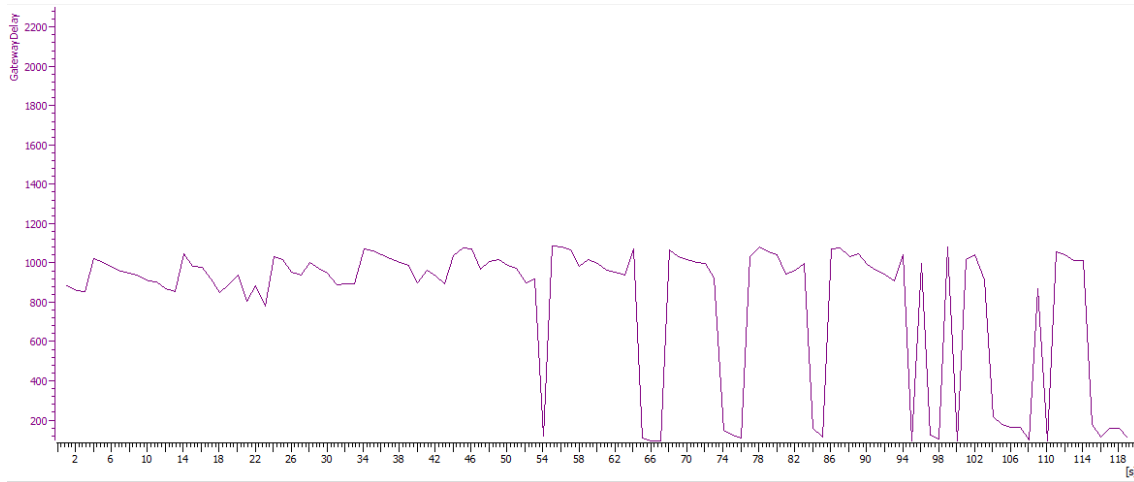


Figure 4.7: Gateway latency measured for the low-priority message during a 120 s multi-signal test. The low-priority message was transmitted at intervals of 1000 ms, corresponding to approximately 120 latency measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at $200 \mu\text{s}$. Consecutive measurements are connected by line segments.

Figure 4.8 and Figure 4.9 show the corresponding measurements for a transmission interval of 100 ms. Compared to the 1000 ms case, both the high- and low-priority signals show more frequent latency variations. The high-priority signal remains more concentrated around a specific latency range, while the low-priority signal contains more peaks and irregular fluctuations. This suggests that the effect of additional network traffic becomes more visible as the transmission interval decreases.

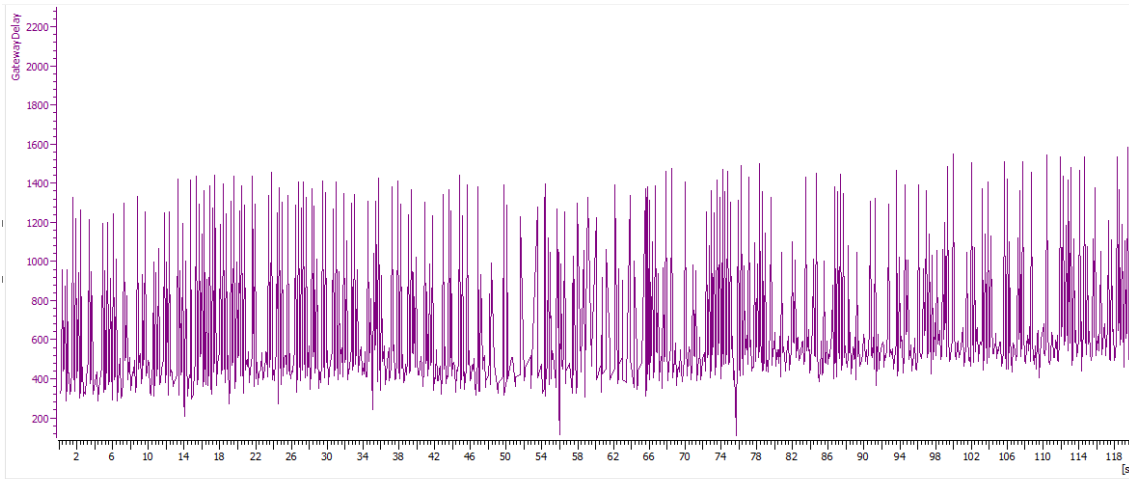


Figure 4.8: Gateway latency measured for the high-priority message during a 120 s multi-signal test. The high-priority message was transmitted at intervals of 100 ms, corresponding to approximately 1,200 latency measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at $200 \mu\text{s}$. Consecutive measurements are connected by line segments.

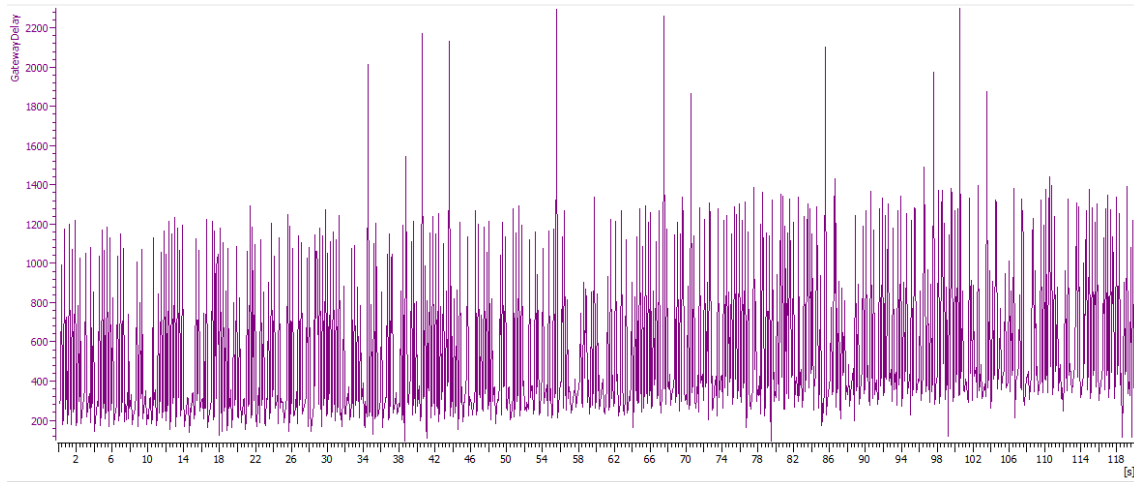


Figure 4.9: Gateway latency measured for the low-priority message during a 120 s multi-signal test. The low-priority message was transmitted at intervals of 100 ms, corresponding to approximately 1,200 latency measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at $200 \mu\text{s}$. Consecutive measurements are connected by line segments.

Figure 4.10 and Figure 4.11 show the measurements for the shortest transmission interval, 10 ms. In this case, both plots become significantly denser due to the higher number of transmitted messages. The high-priority signal still appears more controlled, although frequent variations are present. The low-priority signal shows a wider spread and more irregular behavior, indicating that it is more sensitive to additional traffic and bus load. This result shows that priority becomes more important when the network is exposed to a higher message frequency.

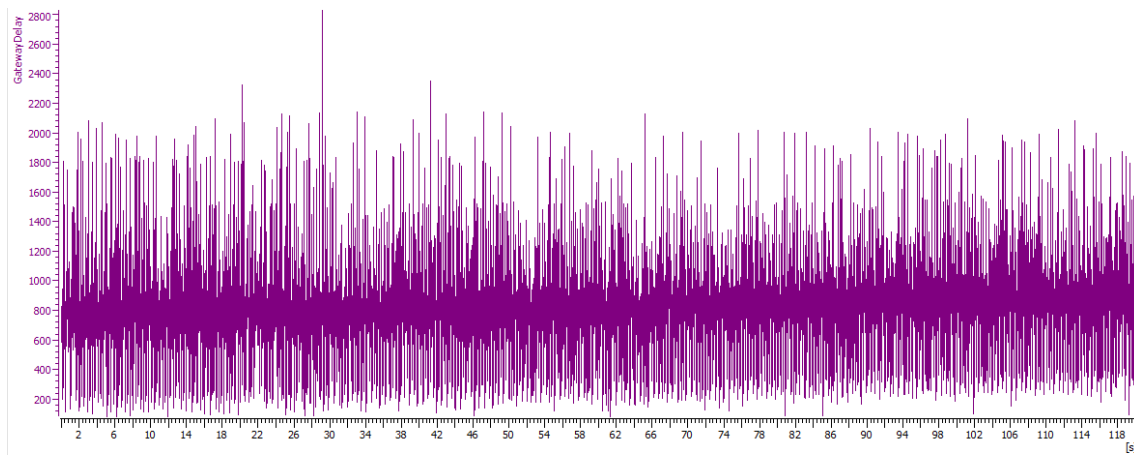


Figure 4.10: Gateway latency measured for the high-priority message during a 120 s multi-signal test. The high-priority message was transmitted at intervals of 10 ms, corresponding to approximately 12,000 latency measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at $200 \mu\text{s}$. Consecutive measurements are connected by line segments.

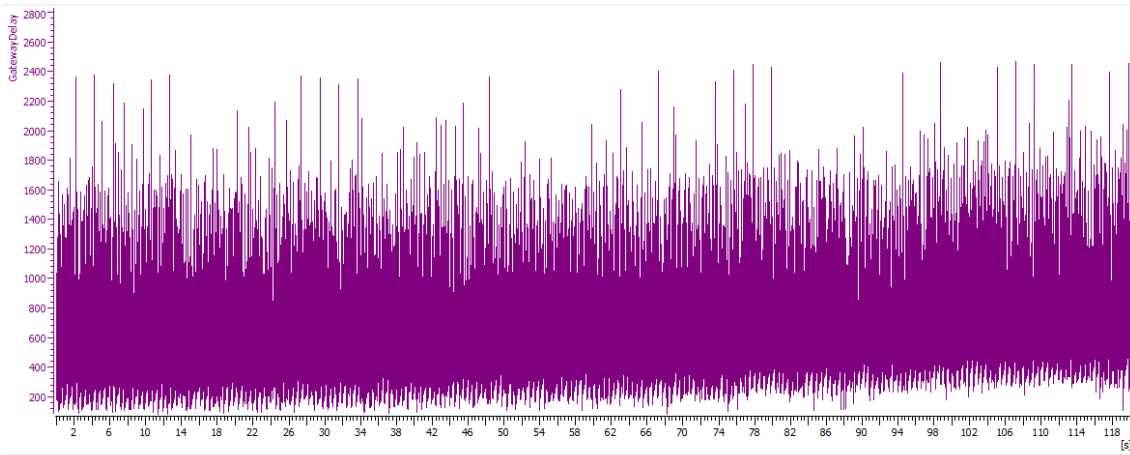


Figure 4.11: Gateway latency measured for the low-priority message during a 120 s multi-signal test. The low-priority message was transmitted at intervals of 10 ms, corresponding to approximately 12,000 latency measurements. Time is given in seconds and latency in microseconds. The vertical axis starts at 200 μ s. Consecutive measurements are connected by line segments.

CAPL in CANalyzer was also used to collect statistical values for the multi-signal tests. For each transmission interval, one analyzed message was tested, and the total number of samples, average latency, minimum latency, and maximum latency were recorded for each case. The results are summarized in Table 4.2.

Test case	Priority	Samples	Average latency	Minimum latency	Maximum latency
1000 ms	High	119	426.09	197.39	596.74
1000 ms	Low	119	806.30	87.63	1086.32
100 ms	High	1199	623.48	109.72	1586.93
100 ms	Low	1199	533.59	94.59	2300.97
10 ms	High	11999	849.25	84.03	3604.62
10 ms	Low	11999	652.44	84.00	2469.56

Table 4.2: Measured gateway latency statistics in microseconds for high- and low-priority messages under multi-signal load conditions.

Multi-signal short discussion

The multi-signal tests show that the measured latency is affected by the presence of additional network traffic. In theory, high-priority CAN messages should have lower latency than low-priority messages, since arbitration gives them earlier access to the network when multiple messages attempt to transmit simultaneously. However, this expected relation is not observed in all cases. In both the 10 ms and 100 ms tests, the high-priority message has a higher average latency than the low-priority message, which is contrary to the expected effect of CAN arbitration.

This indicates that the measured latency is not only determined by CAN priority, but also by other factors in the test setup, such as gateway processing, buffering, message scheduling, CANalyzer timing, or general measurement variation.

The same limitation applies to the multi-signal measurements. Including the median would have provided a more representative comparison between the high- and low-priority signals, particularly in cases containing large outliers.

4.4 Gateway latency in mathematical model

The latency measurements presented in Section 4.3 were used to define a representative gateway delay value for the mathematical optimization model. In the model, gateway delay is treated as an additional delay. This means that every active gateway in the optimized architecture contributes to the total end-to-end delay.

The experimental setup measured the total latency through two gateways, which are CAN-to-AUTBUS gateways. Therefore, the measured latency values represent the combined delay of both gateway conversions, together with the small propagation and measurement-related delays in the physical setup. Since the cable lengths in the test setup were short, the propagation delay contribution is assumed to be negligible compared with the measured gateway latency. The measured value is therefore used as an approximation of the total delay introduced by the two gateways. The measured latency is shown in Table 4.3.

From \ To	CAN	100BASE-T1	1000BASE-T1	AUTBUS
CAN	–	1.0	1.5	1.2
100BASE-T1	1.0	–	0.5	0.8
1000BASE-T1	1.5	0.5	–	0.8
AUTBUS	1.2	0.8	0.8	–

Table 4.3: Gateway delay matrix for conversion between different communication protocols in microseconds.

The gateway delay parameters, measured in microseconds (μs), are included to capture the additional time required for a signal to enter a gateway, be processed and translated, and then leave the gateway in another protocol format.

To obtain a controlled reference value for the optimization model, the average latency from the 100 ms one-signal test was used. The purpose of the gateway-delay parameter d_{PLPR}^{gw} is to represent an effective delay for one gateway conversion under controlled communication conditions, rather than the complete traffic-dependent message latency of a loaded network.

The one-signal test reduces the influence of competing messages, CAN arbitration, traffic-dependent buffering, and scheduling. This makes it more suitable for selecting a fixed reference value for d_{PLPR}^{gw} , since the mathematical model does not explicitly represent individual messages, message priorities, queueing, or time-varying network utilization. The 100 ms case was selected because it represents the interme-

diated transmission interval among the three tested message rates. The selected value should not be interpreted as a universal gateway delay for all operating conditions.

The measured average latency for the 100 ms one-signal test was $304.34 \mu\text{s}$. The experimental setup measured the combined latency through two gateway conversions. Since d_{PLPR}^{gw} represents the delay associated with one gateway conversion, the measured latency was divided equally between the two conversions. This gives an estimated delay of $304.34/2 = 152.17 \mu\text{s}$ per gateway, which was rounded to $152 \mu\text{s}$ for use in the optimization model.

This division is a modelling approximation. It assumes that the two gateway conversions contribute equally to the measured latency. The resulting value should therefore be interpreted as an effective per-gateway delay for the architectural model rather than as an isolated measurement of the internal processing time of one gateway.

The multi-signal results presented in Table 4.2 were not used directly to select the fixed value of d_{PLPR}^{gw} . These measurements include the effects of additional message traffic, arbitration, message priority, and gateway buffering. Directly assigning the complete multi-signal latency to one gateway conversion would therefore incorporate the conditions of one specific traffic scenario into a model that does not explicitly represent these traffic-dependent processes.

Nevertheless, the multi-signal results provide important information about the variation in latency under more realistic network conditions. The measured average total latencies in the multi-signal tests range from $426.09 \mu\text{s}$ to $849.25 \mu\text{s}$. Assuming that each gateway contributes half of the measured latency, these measurements correspond to effective per-gateway delays ranging from approximately $213.05 \mu\text{s}$ to $424.63 \mu\text{s}$.

The difference between this range and the selected value of $152 \mu\text{s}$ demonstrates that gateway-related latency depends on the traffic and measurement conditions. Therefore, the optimization results obtained using $152 \mu\text{s}$ for d_{PLPR}^{gw} should be interpreted as controlled reference-case results and not as predictions of latency under every possible network load.

Before a real-world implementation, the optimization should be complemented by a sensitivity analysis. The values assigned to d_{PLPR}^{gw} should be varied around $152 \mu\text{s}$ and across the range indicated by the multi-signal measurements. The optimization should then be repeated to investigate whether changes in the assumed gateway delays affect the selected topology, the number and placement of gateways, the total network cost, or the resulting end-to-end delay.

Maximum measured latency values should be investigated separately as conservative scenarios. The largest total latency observed in the multi-signal tests was $3604.62 \mu\text{s}$. Using the same approximation, this corresponds to an effective delay

of approximately $1802.31 \mu\text{s}$ per gateway. However, this is an observed maximum from a finite measurement period and should not be interpreted as a guaranteed worst-case bound.

For the architectural comparisons performed in this thesis, $152 \mu\text{s}$ is used consistently as the controlled reference value of $d_{p_L p_R}^{\text{gw}}$. The range obtained from the multi-signal tests and the maximum observed value are retained as important considerations for sensitivity analysis and validation before practical implementation.

5

Numerical Experiments and Results

This chapter presents the numerical results obtained by applying the optimization framework described in Chapter 3 to a representative bus configuration. The chapter begins with a description of the case study and the parameter values used. The Pareto front is then presented and discussed, followed by an analysis of the network topology of three selected Pareto points. The improvement obtained by the fine-tuning model is quantified, and the computational performance of the solver is reported. The chapter concludes with a sensitivity analysis on the most important model parameters.

5.1 Computational setup and data

The cable cost depends on the communication technology used. Since the cost is given per meter, the total cable cost increases with cable length, and different technologies result in different total costs.

Technology	Cost (SEK/m)
CAN	4.2
100BASE-T1	50
1000BASE-T1	50
AUTBUS	4.5

Table 5.1: Cable cost for CAN, 100BASE-T1, 1000BASE-T1, and AUTBUS.

Whenever a signal passes between two different technologies, a gateway must be inserted in between. The gateway cost represents the cost in SEK of converting from protocol p_1 to protocol p_2 .

From \ To	CAN	100BASE-T1	1000BASE-T1	AUTBUS
CAN	-	3450	3619	945
100BASE-T1	3450	-	3999	1020
1000BASE-T1	3619	3999	-	1020
AUTBUS	945	1020	1020	-

Table 5.2: Gateway costs for conversion between the communication protocols.

These parameters describe the cost of using gateways for communication between different technologies. The gateways will increase the total system cost due to additional hardware and design complexity.

5.2 Numerical Experiments

This section evaluates the effects of discretization step size, the ε -constraint method, and fine-tuning on the optimization results. Table 5.3 summarizes the cost reductions achieved through fine-tuning across the generated Pareto frontier.

Point	Orig. cost	FT cost	Δ cost	Δ %	Delay
0	24 900.67	22 809.97	2 090.70	8.40	2.58
1	18 062.20	16 548.27	1 513.93	8.38	10.13
2	13 333.09	12 256.57	1 076.52	8.07	17.69
3	12 179.37	11 252.12	927.25	7.61	25.28
4	9 884.60	9 282.17	602.43	6.09	32.81
5	8 127.20	7 427.43	699.77	8.61	40.41
6	6 039.05	6 033.29	5.76	0.10	48.09
7	5 092.19	5 084.18	8.01	0.16	55.16
8	5 090.22	5 083.58	6.64	0.13	63.02
9	5 088.30	5 083.01	5.29	0.10	70.49
10	4 141.53	4 136.39	5.14	0.12	77.95
11	4 139.39	4 135.79	3.60	0.09	85.81
12	4 137.50	4 135.22	2.28	0.06	93.28
13	4 135.34	4 128.32	7.02	0.17	101.14
14	4 133.36	4 127.75	5.61	0.14	108.61
15	4 131.53	4 127.18	4.35	0.11	116.07
16	4 129.45	4 126.58	2.87	0.07	123.93
17	4 127.61	4 126.01	1.60	0.04	131.40
18	4 125.71	4 121.24	4.47	0.11	138.87
19	4 124.00	4 120.64	3.36	0.08	146.73

Table 5.3: Fine-tuning impact across the Pareto frontier. All costs are given in SEK and delays in milliseconds.

5.2.1 Test 1: Varying the Step Size

In this test, the objective was chosen to minimize the cost. The step size governs how finely the bus is discretised into intervals before optimisation. A smaller step size Δs gives the optimiser more candidate boundary positions, and therefore more freedom to place protocol transitions and gateways, but it also increases the number of binary decision variables and hence the time complexity of the resulting mixed-integer program. To identify a suitable value, the full pipeline was solved at four step sizes, $\text{step_size} \in \{5.0, 2.0, 1.0, 0.5\}$ m, with all other parameters held fixed. For each run the minimum achievable cost, the number of binary variables, the final optimality gap, the termination status, and the solve time were recorded. The results are summarized in Table 5.4.

Step size	Binary vars	Min. cost	Gap (%)	Status	Solve time
5.0	1 249	4 125.18	0.00	Optimal	0.39 s
2.0	3 229	4 124.00	0.00	Optimal	4.4 s
1.0	6 300	2 243.72	0.00	Optimal	3.5 s
0.5	12 589	4 124.00	33.16	Interrupted	~9 h

Table 5.4: Effect of step size on model size, solution quality, and solve time for the cost-minimisation problem. Step size is given in meters, cost is given in SEK, and time is given in seconds unless otherwise stated.

Tractability

For all step sizes of 1 m or coarser, the cost-minimisation problem was defined by the objective in (3.16) was solved to proven optimality (0 % gap) in under five seconds. At stepsize 0.5 m, however, the number of binary variables rose to 12 589 and the problem became intractable: the solver explored over 1.16 million branch-and-bound nodes and 162 million simplex iterations across roughly nine hours of computation, yet terminated at a 33.16 % optimality gap when manually interrupted. Examination of the solver log shows that the best incumbent was already reached after approximately 222 seconds and barely improved thereafter; the remaining time was spent attempting to *prove* optimality rather than finding better solutions. A clear tractability cliff therefore exists between $\text{step_size} = 1$ m and $\text{step_size} = 0.5$ m, establishing 1 m as the practical lower bound on discretisation granularity for this problem instance.

Solution Quality

The minimum cost does not decrease smoothly as the grid is refined. Between 5 m and 2 m the optimal cost is essentially unchanged ($4\,125.18 \rightarrow 4\,124.00$ SEK), but between 2 m and 1 m it drops sharply from 4 124.00 to 2 243.72 SEK, a reduction of approximately 46 %. This non-monotone, cliff-like behaviour indicates that solution quality is sensitive not to raw grid density alone, but to whether the grid aligns with critical positions along the bus, in particular the ECU locations, the manually

specified split points, and the maximum-CAN-length constraint. At 1 m resolution a boundary configuration becomes available that unlocks a substantially cheaper topology (a longer, low-cost AUTOBUS span), whereas the coarser 2 m and 5 m grids cannot place boundaries at the positions required to realise it.

It should be noted that the fine-tune step cannot recover this improvement on its own: fine-tuning optimises interval *lengths* within a *fixed* topology and cannot introduce the new protocol arrangement that the finer grid makes reachable. The discretisation resolution therefore performs genuine work in determining which topologies the optimizer can discover, independent of the subsequent boundary refinement.

Conclusion

Taken together, the two effects identify `step_size = 1 m` as the recommended setting for this problem instance: it is the finest discretizations at which the cost-minimization model solves to proven optimality in practical time, and it simultaneously yields a large ($\sim 46\%$) cost improvement over the coarser grids. Because `step_size = 0.5 m` does not converge in a reasonable time, the true optimum for a step size below 1 m cannot be established; it is bounded above by 2 243.72 SEK but may be lower. This is an inherent limitation of the approach at fine granularity rather than a property of the chosen solution

Taken together, the results identify `step_size = 1 m` as the recommended setting for this problem instance. It is the finest discretization for which the cost-minimisation model was solved to proven optimality within a practical computation time, while yielding an approximately 46 % cost reduction compared with the coarser discretizations.

To improve the branch-and-bound performance for `step_size = 0.5 m`, the optimal solution obtained for `step_size = 1 m`, with an objective value of 2 243.72 SEK, was mapped to the finer discretization and supplied to the solver as an initial incumbent. This allowed the solver to prune nodes whose lower bounds could not improve upon the incumbent solution. The 0.5 m model was run for [TIME], resulting in a best feasible objective value of [OBJECTIVE] SEK and a final optimality gap of [GAP] %. Including the 3.5 s required to obtain the 1 m incumbent, the total computation time was [TOTAL TIME].

Although the finer model [was/was not] solved to proven optimality, these results show that providing a high-quality incumbent substantially improves the computational performance. Nevertheless, `step_size = 1 m` remains the recommended setting because it provides a proven optimal solution with considerably lower computational effort.

5.2.2 Test 2: Sweeping ε

In this section we present the results from employing the ε -constraint method, described in Subsection 1.5.2. As seen in Figure 5.1 and Figure 5.2, the Pareto frontier exhibits a staircase structure with a small number of significant cost reductions separated by flat plateaus. For delay values beyond approximately 38 ms, the cost

remains essentially constant. The delay constraint is no longer binding and the cost-minimising all-CAN topology is already feasible. The non-dominated points in this region satisfy only weak Pareto optimality, since no solution exists that strictly improves both objectives simultaneously, yet accepting additional delay yields no meaningful cost reduction.

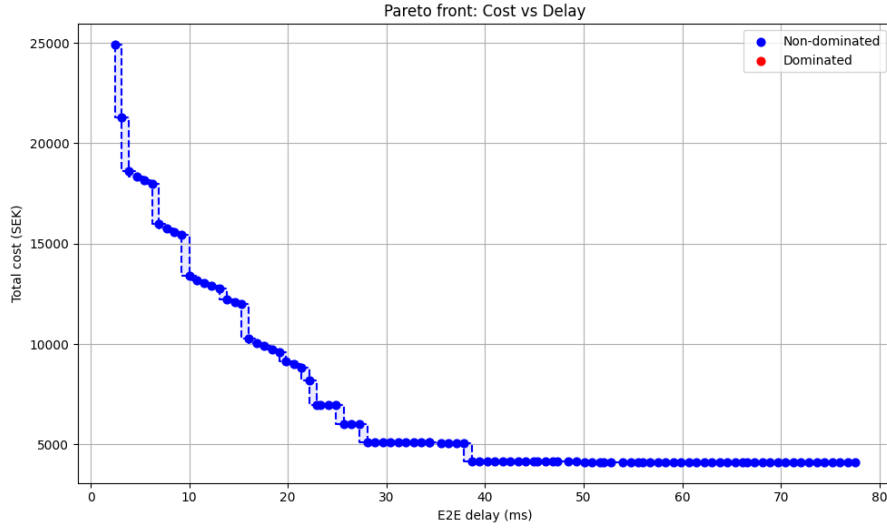


Figure 5.1: Pareto frontier with 100 ε -values.

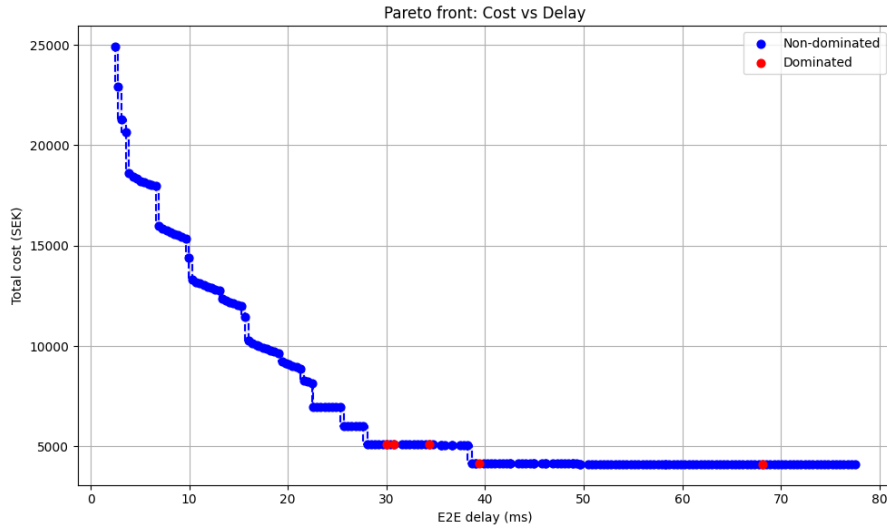


Figure 5.2: Pareto frontier with 200 ε -values

The practically interesting trade-offs occur in the region below approximately 38 ms, where each significant drop in cost corresponds to a structural change in the optimal topology, such as the removal of a gateway or a transition from a high-speed Ethernet segment to a lower-cost protocol. These discrete jumps reflect the combinatorial

nature of the MILP. The cost function can only decrease at points where the integer solution changes, and remains constant between such transitions.

Pareto Frontier and Fine-Tuning Results

Figure 5.3 provides a representation of the Pareto-frontier structure observed in the denser ε -sweeps shown in Figure 5.1 and Figure 5.2. Its purpose is to illustrate the staircase-shaped structure of the frontier and the trade-off between cost and end-to-end delay. The figure is intended to illustrate the qualitative behaviour of the Pareto frontier and should therefore not be used for direct numerical comparison with Figure 5.1 and Figure 5.2, which provide the quantitative results of the ε -sweeps.

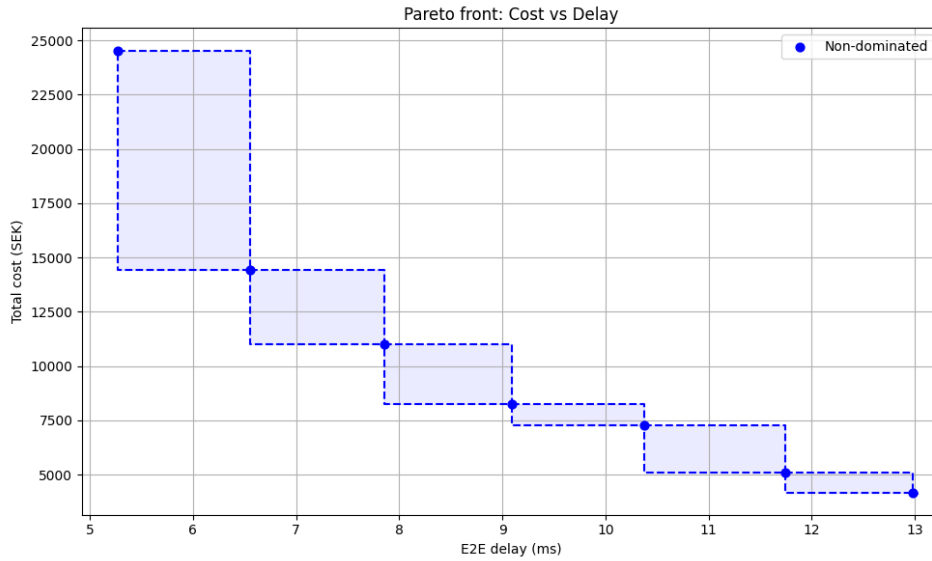


Figure 5.3: Pareto-frontier structure observed in the denser ε -sweeps.

The simplified representation highlights the discrete nature of the optimization problem. As the allowable delay is relaxed, the optimal cost decreases in distinct steps rather than continuously. Each major step corresponds to a change in the selected network topology, for example, through a change in protocol assignment or gateway configuration. The denser sweeps in Figure 5.1 and Figure 5.2 provide the quantitative results, while Figure 5.3 is used to illustrate this staircase behavior more clearly.

Point	Delay (ms)	ILP cost (SEK)	FT cost (SEK)	Δ cost (SEK)
1	5.20	25 000	22 000	3 000
2	6.40	14 700	12 600	2 100
3	7.70	11 000	10 100	900
4	9.00	8 200	7 200	1 000
5	10.30	7 200	6 500	700
6	11.74	5 090	5 081	9
7	12.98	4 140	4 135	5

Table 5.5: Selected ILP solutions and their corresponding fine-tuned solutions. Delay is given in milliseconds, and all costs are given in SEK.

Figure 5.4 shows the selected solutions together with their corresponding fine-tuned solutions overlaid in orange. For the selected solutions used in the fine-tuning analysis, the fine-tuning LP optimizes the interval boundary positions continuously while holding the discrete network architecture fixed. The fine-tuned solutions generally achieve a lower cost than their corresponding ILP solutions, showing that continuous adjustment of the interval boundaries can improve the cost while preserving the selected topology and satisfying the corresponding delay constraint.

Table 5.5 illustrates the effect of fine-tuning for the selected set of solutions by comparing the cost before and after continuous boundary adjustment. The largest absolute improvement among these solutions is observed at point 1, where the fine-tuned cost decreases from 25 000 SEK to approximately 22 000 SEK, corresponding to a reduction of roughly 3 000 SEK. For the lower-cost solutions, the improvements are considerably smaller. At point 6, the cost decreases from 5 090 SEK to 5 081 SEK, while at point 7 it decreases from 4 140 SEK to 4 135 SEK. The corresponding delay values remain unchanged at 11.74 ms and 12.98 ms, respectively. This indicates that the potential for additional cost reduction through boundary repositioning depends on the selected network topology, with some solutions providing considerably more room for improvement than others.

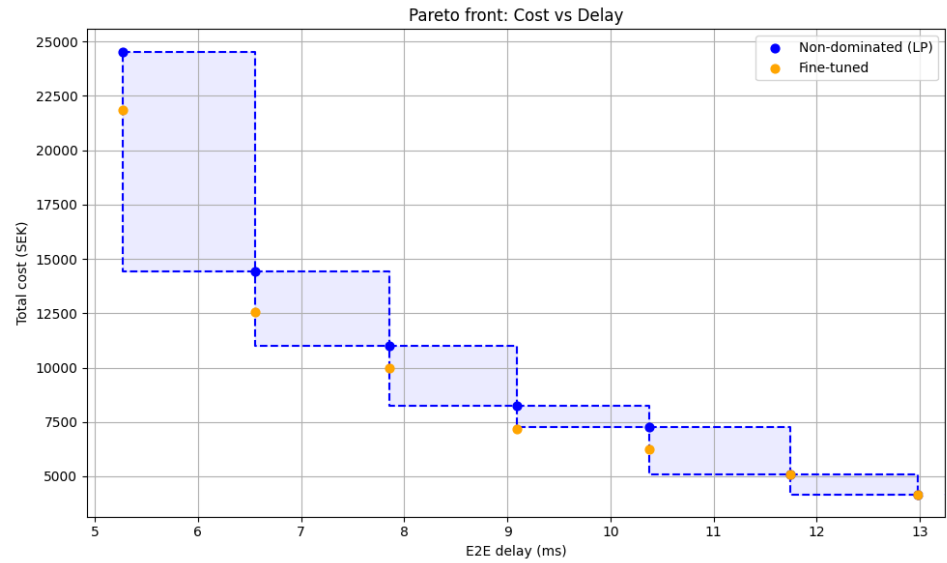


Figure 5.4: Selected ILP solutions with their corresponding fine-tuned solutions overlaid in orange.

6

Discussion

The results presented in this thesis show that the design of a multi-layered in-vehicle communication architecture involves a clear trade-off between communication performance, implementation cost, and computational complexity. Both the physical measurements and the optimization results support the central motivation of the work: long vehicle platforms cannot be evaluated only from a wiring perspective, since communication delay, gateway placement, protocol compatibility, and practical implementation constraints all influence the final architecture.

6.1 Test measurement discussion

The physical tests were performed to characterize the latency introduced when a CAN signal is transmitted through a gateway-based communication path. The single-signal measurements provide a useful baseline, since the analyzed message was transmitted without additional network traffic. Even under these relatively simple conditions, the measured latency was not constant. Instead, the results showed clear variations over time, including several latency peaks. This indicates that the end-to-end delay is not only determined by the physical cable path or by a fixed gateway processing time. Other effects, such as internal gateway buffering, scheduling, CANalyzer timestamping, and general measurement variation, may also contribute to the observed behavior.

Among the one-signal cases, the 10 ms transmission interval resulted in the highest average latency and the most frequent visible fluctuations. The 100 ms case produced the lowest average latency, while the 1000 ms plot appeared less dense and easier to follow because fewer measurements were displayed over an equally long measurement period. These results suggest that the transmission interval may influence both the measured latency and its variability. However, the three test cases do not demonstrate a simple linear relationship between transmission interval and gateway latency. In particular, the 1000 ms case did not produce the lowest average latency, indicating that other timing effects in the experimental setup may have influenced the results. The measured latency should therefore be treated as a statistical quantity rather than a deterministic constant. Consequently, any real-world implementation of the mathematical model should include sensitivity analyzes in which the gateway-delay values are varied across plausible ranges derived from the measurements. More extensive verification of the physical tests would also be required to assess the repeatability and reliability of the measured values, for example,

through repeated experiments, longer measurement periods, additional network-load conditions, and independent verification of the measurement method.

The multi-signal tests further demonstrate that network traffic influences latency behavior. When several messages were transmitted simultaneously, the analyzed signal showed larger variations and higher maximum latency values compared with the one-signal baseline. This is expected, since CAN communication is based on shared bus access and message arbitration. When several messages compete for transmission, waiting time may increase, especially for messages with lower priority identifiers. The results therefore confirm that gateway latency cannot be studied independently from the surrounding communication load. In a real in-vehicle network, where many ECUs transmit messages with different periods and priorities, the effective end-to-end delay will depend on both the gateway path and the bus traffic conditions.

At the same time, the multi-signal results also show that priority alone does not fully explain the observed latency. In theory, high-priority CAN messages should experience lower latency than low-priority messages because they win arbitration when several ECU's attempt to transmit simultaneously. This tendency is visible in some cases, but not consistently across all tests. For example, in the 10 ms multi-signal case, the high-priority message showed a higher average latency than the low-priority message. This unexpected result suggests that the measured delay is affected by more than classical CAN arbitration. Possible explanations include gateway-internal scheduling, buffering effects, message timing alignment, interaction between periodic signals, or limitations in the measurement method. This is an important observation, since it shows that practical gateway behavior may deviate from simplified theoretical assumptions.

The latency measurements are also relevant to the mathematical model, in which gateway delay is included as one component of the total end-to-end delay. However, the experimental results show that gateway delay is not constant in practice, but varies with the transmission interval, network traffic, and possibly the internal state of the gateways. The gateway-delay values used in the optimization model must therefore be selected carefully. Although an average value may be suitable for a nominal analysis, it may underestimate the delay under unfavorable operating conditions. For safety-critical or timing-critical communication paths, more conservative values, such as measured maximum latencies or estimates derived from several test scenarios, may therefore be appropriate. Higher assumed gateway delays may change the network configurations obtained when computing the points on the Pareto front, including their fine-tuned versions. In particular, the model may favor configurations with fewer gateways or faster communication technologies, which may result in a higher total network cost. Here, the increased cost refers to the cost of the resulting network configuration, not to the computational cost of solving the optimization problem.

6.2 Mathematical model discussion

The optimization results show that the proposed model can generate meaningful architecture alternatives by balancing cost and delay. The Pareto-based formulation is suitable for this problem because there is no single universally optimal design. A low-cost architecture tends to use cheaper communication technologies and fewer gateways, but this increases delay. A low-delay architecture instead requires faster technologies or more protocol transitions, which increases the total cost. The Pareto frontier therefore provides a structured way to compare design alternatives. Rather than selecting one solution blindly, the decision maker can select a fine-tuned solution derived from a point on the Pareto front, depending on the acceptable compromise between cost and timing performance.

The fine-tuning results show that additional improvement can be obtained after the initial discretized optimization. For Pareto points 0–5, which are located at the low-delay, high-cost end of the Pareto front, the fine-tuning step reduces the cost by approximately 6–9%. This indicates that the initial discretization can produce a topology that is structurally good but not perfectly positioned in terms of boundary placement. By allowing interval lengths to vary continuously while keeping the protocol sequence fixed, the fine-tuning model can reduce unnecessary use of expensive technologies and improve the cost efficiency of the chosen topology.

However, the cost effect of fine-tuning becomes much smaller for the Pareto points located in the low-cost, high-delay region of the frontier. For these points, the cost reduction is often below 0.2%. This suggests that the corresponding network configurations are already close to the cheapest feasible arrangement, leaving little room for further cost reduction through continuous adjustment of the boundary positions. The fine-tuning model is therefore particularly useful when the selected Pareto point contains longer segments of more expensive communication technologies or several protocol transitions.

The fine-tuning model should be viewed as a refinement of a configuration obtained from the original mixed-integer model, rather than as its replacement. It cannot change the protocol sequence or introduce or remove gateways; it can only adjust the interval lengths and boundary positions within the selected topology.

A complementary fine-tuning approach could minimize delay while limiting the cost to the value of the selected Pareto point. Since the fine-tuning model is a linear programming problem and therefore convex, the ε -constraint method is not necessary at this stage. Instead, a weighted sum of cost and delay could be minimized repeatedly using different weights, while requiring that neither cost nor delay exceeds the corresponding value of the original Pareto point. Varying the weights would generate a piecewise-linear local Pareto frontier for the fixed topology. This would identify different fine-tuned solutions that improve cost, delay, or both relative to the original Pareto point, as illustrated in Figure 5.4.

Another possible extension would be to alternate between the mixed-integer model and the linear fine-tuning model. The topology obtained from the mixed-integer model could first be fine-tuned, after which the refined interval lengths and boundary positions could be used as input to another mixed-integer optimization. The resulting topology could then be fine-tuned again. Repeating this procedure could potentially produce further improvements, although the reductions in cost and delay would likely become smaller in each iteration. In addition, using a denser set of ε values in the initial mixed-integer optimization could identify additional discrete topologies and provide more starting points for fine-tuning. These extensions were not implemented in this thesis but constitute relevant directions for future work.

The step-size analysis highlights one of the most important modelling trade-offs in the thesis. A smaller discretization step gives the optimizer more possible boundary positions and can therefore improve solution quality. This was clearly shown by the cost reduction observed when reducing the step size from 2 m to 1 m. The 1 m discretization enabled a significantly cheaper topology that was not available on the coarser grids. This confirms that discretization resolution is not only a numerical setting, but directly affects which architectures the model can discover.

At the same time, the 0.5 m case demonstrates the computational limitation of using too fine a discretization. The number of binary variables increased substantially, and the solver was unable to prove optimality within practical time. Although a feasible solution was found, the remaining optimality gap was large when the run was interrupted. This shows that there is a tractability limit where additional resolution no longer provides practical benefit because the model becomes too difficult to solve. For the studied case, a step size of 1 m appears to be a reasonable compromise. It provides a substantially better solution quality than coarser grids while still allowing the solver to prove optimality quickly.

These results have practical implications for vehicle architecture design. The results suggest that mixed-technology communication architectures can reduce delay-related limitations, but the benefit depends strongly on where protocol transitions are placed and how many gateways are required. Introducing a faster technology is not automatically beneficial if the added gateway cost, gateway delay, and architectural complexity outweigh the timing improvement. Therefore, the value of the optimization framework lies in its ability to evaluate the complete architecture rather than individual segments in isolation.

The study also shows that experimental validation is necessary when modelling gateway-based communication. The mathematical model provides a systematic and reproducible way to compare architecture alternatives, but it depends on parameter values such as cable cost, gateway cost, propagation delay, transmission delay, and gateway latency. If these parameters are inaccurate or too idealized, the optimized architecture may not represent the true system behavior. The measured latency variation observed in the gateway tests therefore supports the need to combine optimization with physical verification.

6.3 Limitations of the measurement settings and models developed

Several limitations should be considered when interpreting the results. First, the physical tests were performed on a limited gateway setup and do not necessarily capture all conditions present in a complete vehicle network. Second, the measurement period was limited to 120 seconds for each case, which provides useful insight but may not capture rare worst-case timing behavior. Third, the mathematical model simplifies parts of the real architecture, for example, by representing gateway delay with parameter values and by excluding switches for point-to-point Ethernet at ECU boundaries. Another limitation is that electromagnetic interference (EMI) effects were not explicitly included in the model or measurements. Finally, the optimization is performed for a selected bus configuration, meaning that the numerical results may change for another vehicle layout, ECU distribution, or set of technology costs.

Despite these limitations, the results support the main objective of the thesis. The combination of physical testing and mathematical optimization provides a structured method for analyzing long in-vehicle communication architectures. The measurements show that gateway latency is variable and load-dependent, while the optimization results show how cost and delay can be balanced through protocol selection and gateway placement. Together, these results indicate that a multi-layered architecture can be a feasible approach for long bus platforms, provided that gateway behavior, discretization resolution, and practical implementation constraints are carefully considered.

7

Conclusions

This thesis investigated how a multi-layered in-vehicle communication architecture can be modeled and optimized for long bus platforms, with focus on the trade-off between total system cost and communication delay. The motivation for the work was that long vehicle platforms contain distributed ECUs over large physical distances, which increases cable lengths and can create timing-related communication problems, especially for CAN-based networks.

A integer linear optimization model was developed to assign communication technologies to discretized bus intervals and to determine where gateways are required. The model considered CAN, 100BASE-T1, 1000BASE-T1, and AUTBUS, together with cable costs, gateway costs, gateway delay, transmission delay, propagation delay, maximum consecutive CAN length constraint, and multiple other key constraints. Since cost and delay are conflicting objectives, the epsilon-constraint method was used to generate a set of Pareto-optimal architecture alternatives. A second fine-tuning model was also introduced to adjust boundary positions after the discrete topology had been selected.

The results show that the proposed optimization framework can identify meaningful trade-offs between cost and delay. Low-cost solutions generally rely on cheaper communication technologies and fewer protocol transitions, while low-delay solutions require faster technologies and may introduce additional gateway cost and complexity. The Pareto frontier therefore provides a useful basis for comparing different architecture alternatives rather than selecting a single solution without considering the cost-delay compromise.

The cost-minimizing fine-tuning model showed that additional cost reductions can be achieved after the initial discrete optimization, especially for Pareto points in the high-cost, low-delay region. For these configurations, the fine-tuning step reduced the unnecessary use of expensive technology segments while preserving the selected topology and preventing an increase in delay. The cost reduction was smaller for Pareto points in the low-cost, high-delay region because these configurations were already close to the cheapest feasible arrangement. However, this observation applies specifically to fine-tuning with cost minimization as the objective. If delay were minimized instead, while the cost was limited to that of the selected Pareto point, the low-cost points could provide greater opportunities for delay reduction because their initial delay values are higher. The benefit of fine-tuning therefore depends on both the selected objective and the location of the original point on the Pareto

front. In either case, fine-tuning remains a refinement of the topology obtained from the mixed-integer optimization rather than a replacement for that optimization.

The physical gateway measurements showed that gateway latency is not constant. Instead, it varies with transmission interval, traffic load, message timing, and measurement conditions. This means that gateway delay should not be treated as an exact deterministic value in practical vehicle design. In this thesis, an average representative gateway delay was used in the optimization model, but for timing-critical or safety-related communication, a worst-case value may be more appropriate to use.

The step-size analysis also showed that discretization has a direct effect on both solution quality and computational tractability. A finer discretization can make better topologies available to the optimizer, but it also increases the number of binary variables and may make the ILP computationally too demanding. For the studied case, a step size of 1 m provided a practical balance between solution quality and solve time.

A further limitation of the experimental analysis is that median latency values were not included. This was identified only at a late stage of the project, when access to the original measurement setup and the data required for additional analysis was no longer available. Future measurements should therefore include the median as a standard statistical measure alongside the average, minimum, and maximum values, since it is less sensitive to large outliers.

Overall, the thesis demonstrates that mixed-technology communication architectures can be a feasible approach for improving timing performance in long bus platforms. The combination of mathematical optimization and physical gateway measurements provides a structured method for evaluating architecture alternatives while accounting for cost, delay, protocol compatibility, and practical implementation constraints. Future work should include more extensive gateway measurements under realistic vehicle traffic, validation on additional bus configurations, consideration of electromagnetic interference effects, and an extension of the model to add Ethernet switches.

Bibliography

- [1] AUTOSAR, “Requirements on end-to-end (e2e) protection,” https://www.autosar.org/fileadmin/standards/R21-11/FO/AUTOSAR_RS_E2E.pdf, 2019, accessed: 2026-02-27.
- [2] Stellantis, “UN regulation 156: Software update & software update management system (presentation),” <https://fnc.itu.int/wp-content/uploads/2021/03/Kai-Frederik-ZASTROW-STELLANTIS-V2.pdf>, accessed: 2026-02-27.
- [3] S. D. Sudhoff. Lecture 9: Multi-objective optimization. [Online]. Available: <https://engineering.purdue.edu/~sudhoff/ee630/Lecture09.pdf>
- [4] I. Das and J. E. Dennis, “A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems,” *Structural Optimization*, vol. 14, no. 1, pp. 63–69, Aug. 1997. [Online]. Available: <https://doi.org/10.1007/BF01197559>
- [5] W. Lawrenz, Ed., *CAN System Engineering: From Theory to Practical Applications*, 2nd ed. London: Springer, 2013.
- [6] AUTBUS. (2023) Autbus technology. Accessed: 2026-04-28. [Online]. Available: <https://autbus.org/autbus-technology/>
- [7] Vector Informatik, “Introduction to CAN,” 2023, accessed: 2026-02-11. [Online]. Available: <https://elearning.vector.com/mod/page/view.php?id=333>
- [8] B. A. Forouzan, *Data Communications and Networking*, 5th ed. McGraw-Hill, 2013.
- [9] Q. H. Mahmoud, “Performance testing of twisted pair cables,” *Journal of Computer Networks and Communications*, vol. 2008, p. 586427, 2008.
- [10] IEEE 802.3 Ethernet Working Group, “Ieee 802.3 delay constraints,” 2025, default cable velocity factor $n = 0.66$ corresponding to approximately 5 ns/m propagation delay. [Online]. Available: https://www.ieee802.org/3/dm/public/050125/gorshe_3dm_01a_250501.pdf
- [11] Gurobi Optimization, LLC, “Gurobi optimizer reference manual,” 2024. [Online]. Available: <https://www.gurobi.com>
- [12] L. G. Khachiyan, “A polynomial algorithm in linear programming,” *Doklady Akademii Nauk SSSR*, vol. 244, pp. 1093–1096, 1979.
- [13] N. Karmarkar, “A new polynomial-time algorithm for linear programming,” *Combinatorica*, vol. 4, no. 4, pp. 373–395, 1984.
- [14] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman, 1979.
- [15] R. T. Marler and J. S. Arora, “Survey of multi-objective optimization methods for engineering,” *Structural and Multidisciplinary Optimization*, vol. 26, no. 6,

- pp. 369–395, 2004. [Online]. Available: <https://link.springer.com/article/10.1007/s00158-003-0368-6>
- [16] G. B. Dantzig, *Linear Programming and Extensions*. Princeton, NJ: Princeton University Press, 1963.
 - [17] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. Belmont, MA: Athena Scientific, 1997.
 - [18] V. Chvátal, *Linear Programming*. New York: W. H. Freeman, 1983.
 - [19] S. J. Wright, *Primal-Dual Interior-Point Methods*. Philadelphia, PA: SIAM, 1997.
 - [20] A. H. Land and A. G. Doig, “An automatic method of solving discrete programming problems,” *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.
 - [21] L. A. Wolsey, *Integer Programming*. New York: John Wiley & Sons, 1998.

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se



CHALMERS
UNIVERSITY OF TECHNOLOGY