PhD-FSTM-2024-020
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 05/04/2024 in Esch-sur-Alzette

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN Informatique

by

Patrick KELLER
Born on 25 December 1990 in Luxembourg (Luxembourg)

SB-RTPA: Simulation-based Real-Time Performance Analysis using Aggregation of Short Simulations

Dissertation defence committee

Dr Nicolas Navet, dissertation supervisor
Professor, Université du Luxembourg

Dr Thomas Engel, Chairman
Professor, Université du Luxembourg

Dr Ye-Qiong Song, Vice-Chairman
Professor, Université de Lorraine

Dr Frédéric Ridouard
Associate Professor, LIAS - Université de Poitiers

Dr Tingting Hu
Research Scientist, Université du Luxembourg

Abstract

This dissertation introduces a new evaluation paradigm for worst-case traversal times based on aggregation of simulations. As the complexity of real-time networking systems and applications grows, classical methods face increasing challenges. Oftentimes, analytical methods that provide guarantees on the maximal end-to-end latencies are not available, too pessimistic, or too expensive to develop.

The traditional approach to evaluating such systems via simulation requires running long simulations that apply synchronized node start offsets and randomized clock drifts to explore the simulation state space. Running long simulations, however, has significant implications on the design process as they take a long time to be executed, limiting the variations and number of candidate solutions that the designers of such systems can consider. It is observed in this dissertation that long simulations can be rather inefficient as a significant portion of the computational effort is typically spent in states of low interference between flows, which has a reduced probability of causing high end-to-end latencies.

The paradigm proposed by this dissertation aims to mitigate these shortcomings by running many short simulations and aggregating their results. In all test cases applied in this work, end-to-end latencies equivalent to long simulations were observed but for a fraction of the resources. The speedups observed with these simple techniques reached up to a factor of 266, without even considering additional speedups that could be gained via the increased parallelization potential of the approach.

First, the general potential of simply splitting up the long simulations into shorter ones is evaluated, and both synchronized and uniformly sampled node start offsets for running and aggregating these short simulations are investigated. Increases in median latency on a per-flow basis of up to 25.8% are observed for the considered test cases.

To improve the effectiveness and efficiency of the approach, a heuristic to determine an optimized simulation time that maintains the speedup factor is proposed, together with an improved method to sample the node start offsets, namely stratified sampling.

The heuristic to select the simulation time is based on a pretest executed on a small fraction of the total simulation budget. First, half of the budget dedicated to this pretest is used to determine a reference speedup factor. Then, in increasingly smaller slices of the other half of the budget, speedups for increasingly shorter simulation times are determined until they drop below a certain threshold with respect to the reference speedup. This yields the choice of the simulation time.

The improved sampling of node start offsets is based on overlapping stratification over exponentially growing sampling ranges. This approach allows the exploration of diverse solutions while also integrating the high initial interferences observed by the solutions close to the synchronized case, as exploited by the traditional approach. This technique allows a good trade-off between exploration and exploitation of the search space of starting conditions.

Finally, the maximization of the observed end-to-end delays is modeled as a multi-objective optimization Pareto problem. NSGA-II, a popular algorithm to address such problems, is applied. A biased population is initialized based on stratified sampling. The optimization efficiency and effectiveness are then evaluated on two different optimization variations. One variation optimizes only the node start offsets and applies changing random seeds to each simulation to explore the flow orders. The other variation additionally optimizes the flow scheduling order by introducing and adjusting very small frame offsets to control the sending order of flows scheduled simultaneously without impacting the traffic properties.

The evaluation paradigm developed in this dissertation proves to be beneficial for both generating tighter approximations of worst-case traversal times via simulation and reaching results equivalent to long simulations in a small fraction of the simulation time. Additionally, the approach enables the use of highly parallel infrastructure, such as HPCs, as short simulations are independent of each other and can thus be run in parallel.

This opens up paths to exciting future research and applications, including improved optimization and advanced learning methods. It further allows for more responsive and effective design paradigms by exploiting the increased efficiency and parallelization potential, significantly reducing the friction in the typically iterative design process.

Acknowledgments

This manuscript marks not just the destination of years of dedicated research and study but also reflects the unwavering support and encouragement I've been fortunate enough to receive from many individuals along the way. It is with deep gratitude that I take this opportunity to acknowledge their significant contributions.

First and foremost, I want to express my heartfelt thanks to my supervisor and mentor, Prof. Dr. Nicolas Navet. Your guidance and continuous support through the years have been invaluable, not only in shaping this research but also in fostering my growth as a scholar and a person. Your expertise, patience, and belief in my abilities have been instrumental in navigating the challenges of this turbulent journey. As part of my thesis committee, you have provided me with unparalleled support and insights, and for that, I am deeply appreciative.

I also wish to express my sincere gratitude to the other members of the committee, Prof. Dr. rer. nat. Thomas Engel and Dr. Tingting Hu. Your insights, feedback, and rigorous scrutiny have greatly enriched my work, pushing me to reach higher standards. The constructive discussions and your collective wisdom have been pivotal in guiding my research.

A special note of thanks goes to the external jury members, Dr. Jörn Migge, Prof. Dr. Frédéric Ridouard and Prof. Dr. Ye-Qiong Song, who honored me by expressing their interest in and evaluating my work.

On a more personal note, I am unable to find words that can express my gratitude for the moral support and endless encouragement provided by my mother, Josette, my godfather, Jean-Claude, my cousin, Yves and my close friends Chris, Cynthia, Elena, Farrah, Jeff, Laurent, Loris, and Woud. Your unwavering faith in me, during times of doubt, stress, and desperation, has been a source of strength and motivation. All of you have contributed more significantly than you might realize and completing this journey would not have been possible without your support, advice, laughter, and companionship.

In closing, I am profoundly thankful to everyone who has been a part of this journey. This thesis is a reflection of the collective support and encouragement I've received. Thank you for being pillars of strength and sources of inspiration.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Context

As industrial and consumer systems move closer towards full automation, network communication faces increasing complexity and, thus new challenges. Long gone are the days when a car could be serviced without the use of a computer, let alone be functional without one. A modern car today has a multitude of sensors, is permanently connected to the Internet, and may soon be able to drive itself without any intervention of a human being.

All of these advancements come at a cost. In the early days of electronic systems in consumer vehicles, only a handful of integrated processors were needed to make them functional, and their capabilities were usually very limited and specialized to one designated function. Slowly over time, more and more such electronic circuitry was integrated to make vehicles more efficient and offer better comfort to the driver. These additional features range from entertainment to safety systems, such as automated driver assistance. This increased not only the number of options that could be offered to customers but also the resulting complexity and cost of hardware and software systems.

Over time, the number of integrated electronic control units increased from only a handful to over 100 in some premium vehicles. Further, with the introduction of Machine Learning into almost every aspect of our lives and also consumer vehicles, the need for data processing and transmission reaches needs unseen before.

These two factors contribute to the need for efficient, reliable, and scaleable data transmission between integrated processors. In addition to these requirements, with the electrification required by automated driving, the communication between these control units is required to adhere to certain timeliness constraints. Those timeliness constraints are known as real-time requirements and need to be verified at the design time to avoid catastrophic failure at run time. Catastrophic failure in this context ranges from the harmless malfunction of the vehicle to its total destruction and up to the potential cost of life.

It should be obvious that the creation of such complex mission-critical systems cannot be taken lightly and requires significant resources and validation during the design process. However, the design process is usually iterative in nature and is thus slow-paced and resource-hungry due to the increasing complexity of such systems. Small changes can invoke a large number of necessary validations that need to be repeated for every modification.

This is where this dissertation aims to seek improvement by developing a new approach to approximate worst-case traversal times, an important measure in the timeliness of the system, in a generic, quick and cost-efficient manner compared to previous approaches.

1.2 Motivation and Vision

The evaluation of hard real-time network systems can be realized by two types of validation approaches, network simulation and mathematical analysis, like worst-case schedulability analysis, for instance. Analytical methods are typically the preferred option to evaluate and validate real-time networking systems, as they usually provide firm guarantees on the derived worst-case end-to-end delay upper bounds. However, analytical approaches are not always an option, as an analysis may not be available or simply overly pessimistic, which would result in a costly over-dimensioning and under-utilization of the resulting network architecture. This is particularly undesirable if a higher number of instances of the system will be built, like, for instance, consumer vehicles.

Further, developing and optimizing analytical approaches is complicated and costly, and thus not always feasible in practice. In addition, the available analytical approaches struggle to keep up with the increasing complexity, requirements, and diversity of industrial applications. This explosion in complexity stems, among others, from the continuous increase in automation, like, for instance, automated industrial assembly lines or autonomously driving vehicles.

Simulation models, on the other hand, do not provide firm guarantees on the worst-case end-to-end delays but instead produce a lower bound on the actual worst-case delays. The big advantage of simulation approaches is that they are comparably easy to implement for a given system and can closely model the system under investigation without making overly pessimistic assumptions, particularly when different technologies are combined. Further, it was pointed out in recent literature [26] that simulation gains importance over analytical approaches as they struggle to model the increased complexity.

Simulation thus represents a crucial tool in the development of time-critical systems, particularly in industrial applications where the complexity can quickly exceed the scope of analytical approaches, and a scalable and flexible solution is required. Further, it allows the investigation of the behavior of the end system and virtual analysis of specific properties without the need to build a costly prototype first.

In recent years, simulation has gained importance in the design and development of complex embedded real-time systems. However, one of the downsides of simulation is that it is typically slow and resource-hungry to achieve reliable estimates, particularly for increasingly complex systems. Additionally, the traditional approach of running long simulations with specific starting conditions to evaluate networking systems is unsuitable for the highly parallel computing infrastructures widely applied in industry today.

This dissertation addresses both of these shortcomings of simulation for validating worst-case traversal times in real-time Ethernet networks. This is achieved by a new validation paradigm based on the aggregation of short simulations with optimized starting conditions. This allows to more effectively explore the simulation state space, increasing the probability and flexibility to observe higher end-to-end delays more quickly compared to the traditional approach. This results in a significant improvement in both maximally observed delays and the reduction of computational efforts.

The original vision of this thesis was to apply Machine Learning to (partially) automate the design process and assist the human designer of such systems to explore a larger portion of the possible design space and potentially unconventional but improved solutions a human designer would typically not consider.

Machine Learning was applied successfully in the field of real-time networking before, but it became obvious early that some preconditions for achieving the original goal were not met. Specifically, the tools involved in evaluating the configurations generated by the automated design process would be too slow. This would induce friction and excessive resource requirements that would impede both the learning process required by Machine Learning and the design process itself.

Beyond improving the effectiveness of simulation as a validation tool of real-time Ethernet networks, the flexibility introduced by the paradigm developed in this dissertation further paves the way towards enabling such intelligent and automated architecture design approaches.

1.3 Objectives and Applications

This dissertation aims to provide a new simulation-based validation paradigm for end-to-end delays in Ethernet-based real-time networking systems.

The **objectives** of this paradigm are:

- General applicability that is not constrained to certain traffic types, congestion management mechanisms, protocols, or other properties. This is achieved by reducing the embedded expert knowledge to a minimum.
- Support various available features, mechanisms and tools developed for real-time Ethernet standards like TSN, AFDX and TTEthernet. This is achieved by not relying on any properties or features exclusive to specific implementations but rather on the common foundations of real-time Ethernet, typically provided by every network simulation software supporting real-time Ethernet networks.

- Offering a scaleable approach to simulation-based evaluation of Ethernet-based embedded distributed real-time systems. This is achieved by the aggregation of the results of many short simulations that are independent of each other and can thus be run in parallel.

Some **advantages** can be derived based on these objectives. As discussed, due to the nature of the aggregation, the paradigm greatly improves the potential and simplification of parallelization, and thus the flexibility when validating such systems, as aggregated short simulations are independent of each other. Concretely, this novel paradigm provides the following advantages over the traditional simulation approach:

- Improved tightness of observed lower bounds on the worst-case end-to-end delays under equal simulation budget.
- Generation of equivalent bounds with a fraction of invested simulation budget.
- Greatly improved parallelizability by running many short, independent simulations.
- Targeted evaluation of a subset of flows in the network.

The targeted evaluation of a subset of flows is not explicitly investigated in this thesis and will be discussed a bit further in Chapter 6, as it is possible only under certain conditions and is beyond the scope of this dissertation. Various important practical and scientific applications from the domain of real-time networks can be addressed by the presented paradigm:

- Generation of tighter lower-bounds for systems where no or only overly pessimistic analytical approaches are available.
- Improved evaluation of the pessimism of analytical methods by generating tighter lower-bounds via simulation.
- Speeding up the evaluation of candidate architectures during the design phase.
- Enabling improvements for Design Space Exploration of real-time systems via the increased efficiency and flexibility of the aggregation paradigm.

1.4 Summary of the Approach

In this section, a brief outline of the core components of the aggregation approach is presented. The purpose is to give the reader an early insight into the general functioning such that the details can be put into relation easily and a logical flow for the dissertation can be unfolded.

The fundamental idea of the approach is to break long simulations down into shorter simulations and change the starting conditions for each short simulation to allow a more efficient exploration of the simulation space. The main factor to be explored are the node start offsets, which allow to steer how traffic interacts in the network bridges.

To generate diverse but effective starting conditions, exponentially stratified overlapping sampling ranges should be used to determine the node start offsets. This allows for balanced exploration and exploitation by enabling the exploration of starting conditions further away from the synchronous case while also exploiting the high interference patterns as generated by the traditionally applied synchronous case.

The simulation time for these short simulations should be determined depending on the application goal and, thus, depending on the strategy of choice for setting the node start offsets, such as random sampling or optimization. If the goal is to achieve representative end-to-end delays as quickly as possible, random sampling is the preferred strategy to determine the node start offsets. In that case, the simulation time should be chosen as short as possible while maintaining a reasonable speedup factor, which depends primarily on the simulation software and the complexity of the use case.

When maximizing the observed end-to-end delays is the primary goal, the aggregation approach using multi-objective optimization has shown to be superior, based on an optimization algorithm like NSGA-II, for instance. As observed in this dissertation, it can be beneficial in that case to apply simulation times in the millisecond range to further increase the exploration potential required for optimization. When several flows originate from the same node and no frame scheduling order is prescribed, as is typically the case, it can be beneficial to additionally optimize this order. In the context of this dissertation, this is achieved by applying minimal frame offsets that allow breaking ties in scheduling without significantly impacting the traffic characteristics.

1.5 Research Question Overview

Three chapters represent the scientific core content of this dissertation. They are developed and based on three different publications ([56], [54], [55]) that have been prepared and/or published. Each chapter describes and answers a set of research questions according to a certain goal, incrementally analyzing and developing the aggregation approach further. An overarching research question to summarize the goals of this dissertation could be formulated as follows:

- Is there a viable and efficient alternative to the traditional approach of evaluating flow WCTT in real-time Ethernet networks by running long singular simulations from synchronized node start offsets and randomized clock drifts?

The answer to that question is, in fact, ”yes” and is embodied by the approach of aggregating short simulations while using and optimizing individual starting conditions. The efficiency and effectiveness of approximating worst-case traversal times can be improved when these starting conditions are adjusted in a meaningful manner, as will be explored throughout this dissertation.

To develop and evaluate the aggregation approach, three incremental chapters analyze the efficiency and effectiveness of different variations and extensions. Chapter 3 introduces the most basic version of the aggregation approach using node start offsets that are either synchronized, or uniformly sampled from a fixed range. The two variants to select the starting conditions of the short simulations that are aggregated are compared and evaluated based on five configurations derived from industrially relevant use cases. Chapter 4 explores and evaluates how the approach can be improved by applying stratified sampling to determine the node start offsets and by selecting a simulation time that maintains a reasonable speedup factor. Finally, Chapter 5 investigates how optimization of node start offsets and flow scheduling order can be used to further push the boundaries of the observed WCTT using a significantly reduced simulation time, regardless of the speedup factor.

The different research questions according to these chapters can be summarized, in order, as follows:

- Does the aggregation of short simulations with synchronized node start offsets, like done by the traditional approach, yield comparable results?
- Does the aggregation using node start offsets uniformly sampled from a fixed range yield any benefits over synchronized node start offsets?
- What is the importance of simulation time and node start offset range in the aggregation approach?
- What criteria should be considered when selecting the short simulation time?
- How can we improve the choice for node start offsets in order to maximize the performance of the aggregation?
- How well do the proposed methods to determine simulation time and node start offsets work when applied to the test cases?
- How effective is the simulation aggregation approach when minimizing the simulation time regardless of the speedup factor?
- Can the observed traversal times be increased by applying optimization algorithms, like NSGA-II, to optimize node start offsets with and without additionally optimizing the flow scheduling order?
- What is the overhead cost of minimized simulation time and optimization, respectively?

Chapters 3, 4 and 5 address three of those research questions each. They are designed to incrementally develop, improve and evaluate the performance of the proposed simulation aggregation approach variations.

1.6 Contributions

The core contributions of this dissertation can be summarized as follows:

- In-depth understanding of the effects of important simulation parameters.
- Insights into the potential of evaluating WCTT via different variations of the simulation aggregation paradigm.
- Analysis of the advantages of the aggregation approach variations with uniform sampling, stratified sampling, and optimization of the starting conditions of the aggregated simulations.
- Insights that help implement efficient validation of WCTT via simulation in practice, depending on the primary application metric, such as resource-friendly quick evaluation, or increased quality of the determined bounds.
- Aggregation approach as a simple methodology to make the best use of highly parallel infrastructures such as HPCs to increase the versatility and applicability of network simulation in industrial settings.
- A foundation for promising future research directions that may enable the development of modernized and automated design systems for hard real-time distributed cyber-physical systems.

1.7 Outline of the Dissertation

This chapter provided an introduction to the topic, applications, and research goals of the dissertation, a short overview of how the developed methods can be applied, and a summary of the findings and contributions.

Chapter 2 will introduce the network model, relevant terminology and methodology, which serves as a foundation for the dissertation. The following three chapters will each introduce an incremental step in the research evolution of this dissertation, each corresponding to a publication prepared to address each variation.

Chapter 3 introduces and evaluates the performance of the general approach of aggregating short simulations without optimizing any parameters and will give significant context knowledge about insights leading to the development of the approach.

Chapter 4 focuses on understanding the relevance of simulation time and node start offsets, two central parameters to influence the effectiveness of the approach. Methods are proposed to determine these starting conditions based on factors relevant to practical and scientific scenarios.

Chapter 5 investigates the implications of minimizing the simulation time of short simulations and applying optimization algorithms to determine the node start offsets and, optionally, the flow scheduling order. The aim is to exploit shared traffic patterns between flows that lead to overall high end-to-end delays. The potential of optimizing the flow scheduling order in addition to the node start offsets is investigated.

Chapter 6 concludes the results and findings of the dissertation and suggests promising directions for future research based on these findings. Options to further improve the performance of the aggregation approach are discussed and potential ways to improve design automation based on the new paradigm are proposed.

Chapter 2
Performance Evaluation of TSN Networks

This dissertation focuses on modern switched Ethernet-based communication networks that host traffic that is subject to timeliness and high bandwidth requirements.

Classical real-time network communication was primarily based on bus systems. These bus systems usually function such that all nodes communicate through a single interconnected channel, or multiple channels divided by function and criticality. In these systems, only one node can communicate at a given time on a channel, while the others listen. To increase the bandwidth provided by this approach, more channels and, thus, more cables had to be added. This would result in increased weight and cost.

Popular bus systems include CAN-bus, LIN and FlexRay. Those bus systems are rather simple and, thus, comparatively easy to verify. However, they do not scale well to complex modern systems with high numbers of nodes and high bandwidth requirements induced by technological advances in industry and consumer applications. Particularly, the success of machine learning and automation in many different industries has led to increased bandwidth requirements of such systems.

In this chapter, the technologies and methods relevant to this dissertation will be introduced together with details, assumptions and context of the network model.

2.1 Ethernet-based Communication in Real-Time Embedded Systems

Ethernet-based communication was introduced in real-time embedded systems when requirements began to exceed the limitations of bus-based systems like CAN. These technologies could no longer meet the data-rate requirements of increasingly complex real-time systems.

Before the introduction of Ethernet in real-time embedded systems, available data-rates were limited by bus technologies from 20 kbit/s for LIN, 1 Mbit/s for CAN up to a maximum of 10 Mbit/s for FlexRay. In the search for a solution to increase the available data-rates, Ethernet became an attractive candidate to consider as it provides data-rates up to 1000 times higher than the fastest available bus technology. In addition, as it was developed for general computer networks, it provides a flexible, simple, and scalable solution for growing network complexities, and affordable off-the-shelf components are widely available.

The original Ethernet standard defined as IEEE 802.3 was developed in the 1980s for the application in general computer networking. Despite the lack of mechanisms to guarantee timely communication, Ethernet supports a high bandwidth and provides a strong platform for realizing real-time systems. This led to the development of various Ethernet-based standards for real-time networking.

The first emerging real-time standards were proprietary and originally developed for very specific narrow applications. Namely, those standards were AFDX and TTEthernet, which aim to extend the original Ethernet standard with real-time capabilities. These will be discussed in more detail subsequently. Later, a more open and flexible set of standards, known as Time-Sensitive Networking (TSN), was derived from a set of standards, known as Audio-Video-Bridging (AVB). AVB was originally designed to synchronize and improve professional audio-video applications over Ethernet.

In this dissertation, the experiments will be run based on the TSN technology as it can arguably be considered the most flexible and open standard for Ethernet-based real-time networking, and will be described in more depth in this chapter. The aggregation approach proposed in this dissertation does not rely on any properties or features exclusive to a specific Ethernet-based real-time standard.

All of these standards co-exist to this day, and each has its unique advantages, disadvantages, and applications. Additionally, due to their similarities and shared foundation, the technologies can, in principle, be combined together into a single Ethernet network. Finzi et al., for instance, suggest in [41] an integration of TSN/BLS into AFDX.

2.1.1 Avionics Full-Duplex Switched Ethernet Standard

Avionics Full-Duplex Switched Ethernet (AFDX), also known as ARINC 664 part 7 [8], is a proprietary Ethernet-based communication standard. Airbus originally developed it for reliable and deterministic communication inside avionic applications to enable electronic control of airplanes via fly-by-wire. Due to the complexity and requirements of the application, this was not possible with bus-based technology used in previous applications. The approach was later standardized and patented.

The goal in the development of AFDX was to enable hard time-critical communication and reliability via redundancy, while reducing weight and cost. The Ethernet technology provided a significant improvement in data throughput and thus allowed for reduced weight by reducing the wiring needs compared to previous technologies. In addition, Ethernet could be realized by applying mature off-the-shelf components widely available for general computer networks as a foundation. This allowed to maintain a low hardware cost.

To make the Ethernet standard fit for real-time communication, AFDX applies a rate-constrained traffic paradigm. Data transmissions are partitioned into virtual links, each with a dedicated bandwidth to achieve deterministic transmission behavior by traffic policing and Quality-of-Service mechanisms. To support the full AFDX capabilities, for instance, redundancy, specialized hardware needs to be implemented. More in-depth information on the evolution of AFDX is provided by Fuchs et al. in [42].

2.1.2 TTEthernet Standard

Time-Triggered Ethernet or TTEthernet, standardized as SAE AS6802 in [9], was developed by TTTech Computertechnik AG. Its primary goal is to implement mixed-criticality traffic on Ethernet networks, allowing for traffic that is subject to real-time constraints and efficient transmission of non-critical or best-effort traffic.

As the name suggests, the core principle behind Time-Triggered Ethernet is to synchronize traffic across the network by using time-triggered release to guarantee the timeliness and predictability of critical traffic. TTEthernet can support three types of traffic, time-triggered critical traffic, rate-constrained critical traffic and best-effort traffic, which is served according to available resources. The rate-constrained traffic can be understood as an equivalent of how AFDX traffic is handled.

This approach offers some advantages over AFDX but also comes at a certain overhead cost in terms of the requirements for synchronization, increased hardware requirements and implications on traffic properties. The main downside of TTEthernet is that it requires specialized hardware capabilities to synchronize nodes and thus adding another layer of complexity and cost. More in-depth information on TTEthernet is provided in the whitepaper [7] by TTTech Computertechnik AG.

Whether time-triggered or rate-constrained approaches are preferential is a discussion that is beyond the scope of this thesis and was investigated, for instance, by Hotescu et al. in [51].

2.1.3 IEEE 802.1 Time-Sensitive Networking Standard

Time-Sensitive Networking, or TSN for short, is another standard to enable deterministic and reliable real-time networking based on Ethernet technology and is standardized by the IEEE. More precisely, TSN is a set of standards that can be dynamically combined to fit the needs and requirements of the application specifically.

The core set of standards belonging to TSN [2] were originally developed to improve audio and video transmission over Ethernet for professional audio-video applications, such as conference systems and multimedia. The original aim of this set of standards, known as audio-video-bridging (AVB), was to address the lack of a qualitative unified solution across various multi-media domains and applications.

The original standards belonging to AVB include clock synchronization via the gPTP protocol (802.1AS), traffic shaping via credit-based scheduling (802.1Qav) and VLAN tags allowing the introduction of priority levels to traffic (802.1Q), and a stream reservation protocol (802.1at) to reserve resources for the traffic. As the scope of use cases was widened, the original AVB working group was renamed into TSN, and continued improving and extending the standards for more general purpose real-time applications.

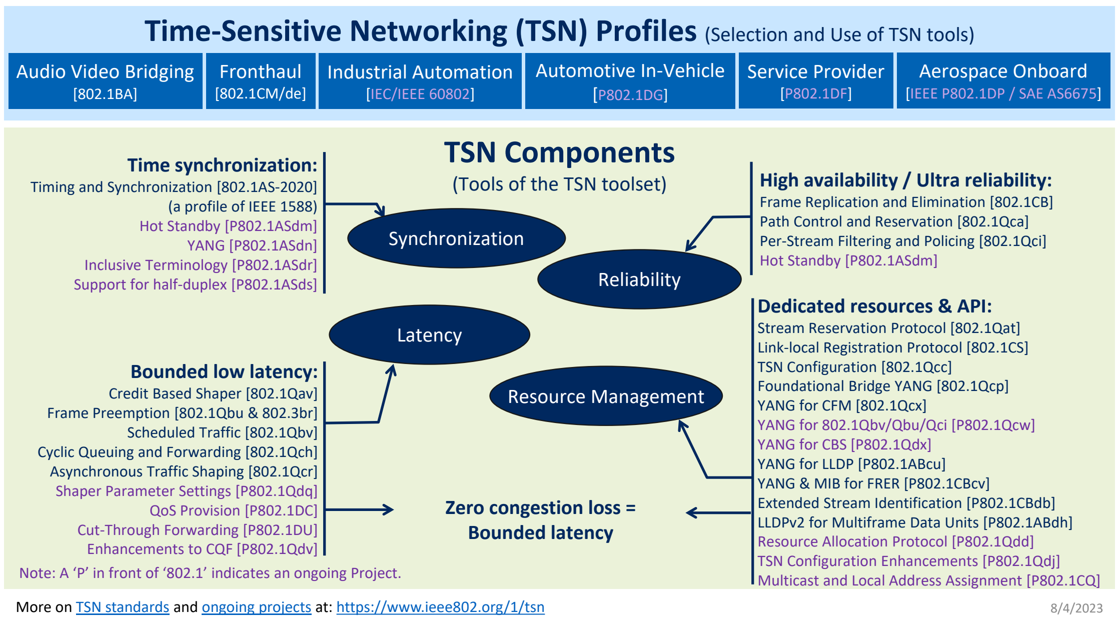

Time-Sensitive Networking (TSN) Profiles (Selection and Use of TSN tools)
Audio Video Bridging [802.1BA]
Fronthaul [802.1CM/de]
Industrial Automation [IEC/IEEE 60802]
Automotive In-Vehicle [P802.1DG]
Service Provider [P802.1DF]
Aerospace Onboard [IEEE P802.1DP / SAE AS6675]
TSN Components
(Tools of the TSN toolset)
Time synchronization:
Timing and Synchronization [802.1AS-2020]
(a profile of IEEE 1588)
Hot Standby [P802.1ASdm]
YANG [P802.1ASdn]
Inclusive Terminology [P802.1ASdr]
Support for half-duplex [P802.1ASds]
Synchronization
Reliability
Latency
Resource Management
High availability / Ultra reliability:
Frame Replication and Elimination [802.1CB]
Path Control and Reservation [802.1Qca]
Per-Stream Filtering and Policing [802.1Qci]
Hot Standby [P802.1ASdm]
Dedicated resources & API:
Stream Reservation Protocol [802.1Qat]
Link-local Registration Protocol [802.1CS]
TSN Configuration [802.1Qcc]
Foundational Bridge YANG [802.1Qcp]
YANG for CFM [802.1Qcx]
YANG for 802.1Qbv/Qbu/Qci [P802.1Qcw]
YANG for CBS [P802.1Qdx]
YANG for LLDP [802.1ABcu]
YANG & MIB for FRER [P802.1CBcv]
Extended Stream Identification [P802.1CBdb]
LLDPv2 for Multiframe Data Units [P802.1ABdh]
Resource Allocation Protocol [P802.1Qdd]
TSN Configuration Enhancements [P802.1Qdj]
Multicast and Local Address Assignment [P802.1CQ]
Bounded low latency:
Credit Based Shaper [802.1Qav]
Frame Preemption [802.1Qbu & 802.3br]
Scheduled Traffic [802.1Qbv]
Cyclic Queuing and Forwarding [802.1Qch]
Asynchronous Traffic Shaping [802.1Qcr]
Shaper Parameter Settings [P802.1Qdq]
QoS Provision [P802.1DC]
Cut-Through Forwarding [P802.1DU]
Enhancements to CQF [P802.1Qdv]
Zero congestion loss = Bounded latency
Note: A 'P' in front of '802.1' indicates an ongoing Project.
More on TSN standards and ongoing projects at: https://www.ieee802.org/1/tsn
8/4/2023

Figure 2.1: **Overview on standards and profiles defined by IEEE 802.1 TSN.** The figure divides the standards according to their role for synchronization, latency, reliability or resource management. Standards that are colored purple or whose acronym starts with a P are still in ongoing development. The figure is provided by the official TSN task group website in [2].

TSN, like AFDX and TTEthernet, is thus an extension to the original switched Ethernet IEEE 802.3 standard. The standards defined by the TSN task group aim to provide reliable, deterministic and time-bounded communication over Ethernet and can be integrated with standard switched Ethernet networks. For that purpose, the set of standards was extended to be applicable to a wide variety of use cases. The core features of TSN include using Ethernet as a transport medium, traffic shaping and synchronization, inherited from the AVB standard it was build upon.

As the TSN standard is based on the Ethernet standard technology that has been developed for decades prior, cost-efficient off-the-shelf hardware is available to build such systems. However, to support certain TSN standards, specialized hardware that provides the necessary capabilities may be required and is consequently more costly. Due to the flexibility of the TSN standards design, the specialized hardware often only needs to be used for the sections of the network where the functionality is required.

TSN today includes a variety of optional standards that can be selected according to the needs of the application and allow tailoring a flexible solution that fits its specific needs while limiting the additional hardware cost to a minimum. Figure 2.1 gives an overview of the available standards divided into four different categories according to their role, such as synchronization, latency, reliability, and resource management. Further, the TSN task group defined several profiles for specific applications, which recommend a subset of standards and configurations. TSN allows for the precise selection and combination of standards as needed for a specific application.

It should again be highlighted that TSN allows to mix hardware that supports certain TSN capabilities as locally required, including Ethernet hardware not specifically developed for TSN, if certain capabilities are not needed in some part of the network. This allows tailoring a very cost-efficient and specialized solution to the needs of an application. In 2021, Seol, Hyeon, Min, Kim, and Paek published a large-scale survey [91] on TSN where they analysed the state-of-the-art developments and trends.

In this dissertation, the focus is on TSN as it arguably provides equivalent capabilities as AFDX and TTEthernet, while providing higher flexibility and potentially less expensive hardware setups. However, even though the simulation experiments are conducted using TSN, the aggregation paradigm does not explicitly rely on TSN-exclusive properties and can thus also be applied to AFDX and TTEthernet.

2.1.4 Ethernet Bus 10Base-T1S

Despite the advantages and general simplicity that Ethernet brings as a network technology, bus-based communication is still required for certain functionality. Consider, for instance, the scenario of connecting ultra-sonic park distance control sensors (PDC) to an automotive TSN network. These sensors have low data-rate requirements and connecting each sensor individually via a 100Mbit/s link would not be a cost-efficient solution.

Previously, to solve this issue, the sensors would be connected via a legacy bus-based zonal network like CAN and communicate to the Ethernet backbone via a gateway. To move towards more homogeneous all-Ethernet networks and away from legacy bus systems, 10Base-T1S (IEEE 802.3cg) was developed. The standard defines a single-pair half-duplex Ethernet bus system that allows to seamlessly integrate with TSN Ethernet networks. It provides a data-rate of up to 10Mbit/s and is thus competitive with the fastest legacy bus systems like FlexRay. In [79], Min et al. describe the functioning and some of the challenges faced by 10Base-T1S.

2.2 Network Model

The network model describes the different components a network consists of and their properties. The scope of this dissertation specifically aims at switched Ethernet networks supporting real-time communication based on the set of TSN standards. Figure 2.2 shows an example of a simple network architecture composed of seven end-nodes and three switches connected via nine links. The different network components and their role and relevant properties will be discussed in more detail next.

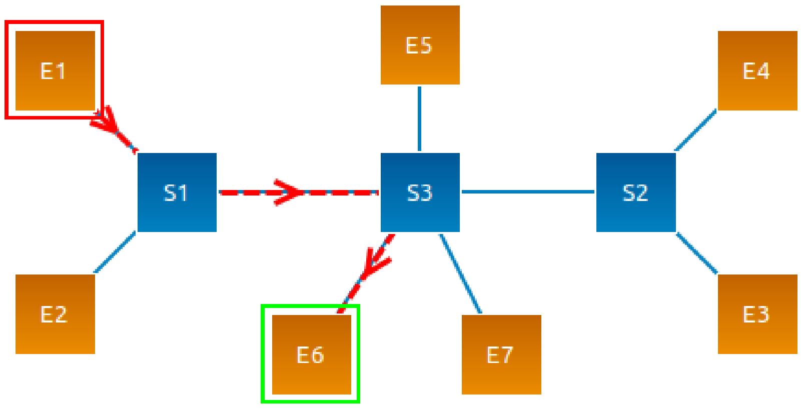


Figure 2.2: **A sample TSN Ethernet network.**
The network connects seven end-nodes denoted by E1 to E7 via three switches denoted by S1, S2 and S3. The end-nodes are displayed as orange squares and the switches are displayed as blue squares. The blue lines represent links that connect end-nodes and switches to each other. Further, the routing of a flow ”v1” is highlighted by a red dashed line with arrows that show the flow direction. The flow is sent from end-node E1 and travels via switches S1 and S3, and is received by end-node E6.

2.2.1 End-Nodes

An end-node is a network-connected device that communicates with other end-nodes connected to the same network. They typically include a processing unit and a network interface, and provide computing, sensing, and/or actuation capabilities. The end-nodes are depicted in Figure 2.2 as orange squares, denoted with names starting with ”E” in the showcased example network.

Depending on the context and their specific role in the system, end-nodes are known under a multitude of different names. In this dissertation, they will typically be referred to as ”end-nodes” or simply ”nodes”. Other names that can frequently be encountered in the literature, depending on the specific field and type of network, include ”network node”, ”device”, ”user device”, ”network device”, ”endpoint”, ”client”, ”host”, ”server”, ”access point”, ”station”, ”peer”, ”terminal” or ”workstation”.

The terms ”node” and ”network node” can refer to either end-nodes or switches. The term is usually applied when speaking of something that can be either of them. For instance, a switch is connected to other network nodes, which can be end-nodes or other switches. When an end-node or switch is referred to specifically, it is denoted as such.

Node Clock Drifts

End-nodes contain a processor clock or a dedicated hardware clock that runs at a certain speed. This clock is usually imperfect and runs slower or faster than the reference system clock. Over time, the node clock drifts away from the reference clock and the relative velocity is known as clock drift. This effect is demonstrated in Figure 2.3 on a periodic flow that is sent from a node, where each trace represents a different (exaggerated) clock drift of -20, 0, and +20 percent.

Clock drifts are influenced by various sources, such as hardware imperfections, aging, temperatures and other environmental factors. While clock drifts are often considered in simulation, some of the factors impacting the drift can change over time, which is typically not considered. Instead, clock drifts are usually modeled as a constant difference in velocity relative to a reference clock.

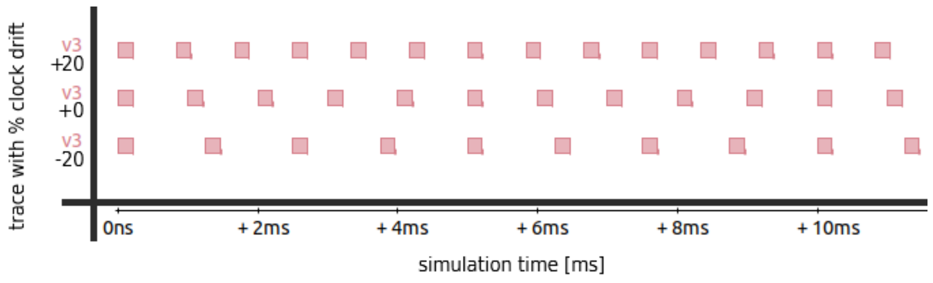


Figure 2.3: **A sample TSN simulation trace of a flow to demonstrate the effect of node clock drifts.**
Three variations of the same periodic flow trace are shown (on the y-axis) for identical network configurations except for changing clock drifts of the sender node. It can be seen that the traffic generated in the same time frame (x-axis) differs for the different clock drifts. To more clearly visualize the effect of clock drifts, extreme clock drifts of -20, 0, and +20 percent were chosen, which is far beyond what is observed in reality. It can be observed that the -20% clock drift causes the clock to run slower and thus produce packets less frequently, while the +20% clock drift version causes the clock to run much faster and thus produce significantly more traffic in the same time frame.

Node Start Offsets

An effect of end-nodes that occurs at the bootup of a system is known as node start offset or NSO for short. It defines a temporal offset from the origin of time, which marks the start of the network system. This offset describes when an end-node is ready, at the earliest, to start sending data. This effect is demonstrated in Figure 2.4, where all end-nodes except for E1 start sending data at time t=0ms, while E1 start sending only at t=0.5ms.

Dataflows originating from an end-node are subject to its node start offset and can only start scheduling and sending data as soon as the starting time has elapsed. In a real system, these node start offsets are subject to different factors and fluctuations and cannot be controlled or determined precisely in general.

These NSOs are an element of major interest in this thesis and will be one of the core parameters that will be explored in order to generate high traversal times in short simulations.

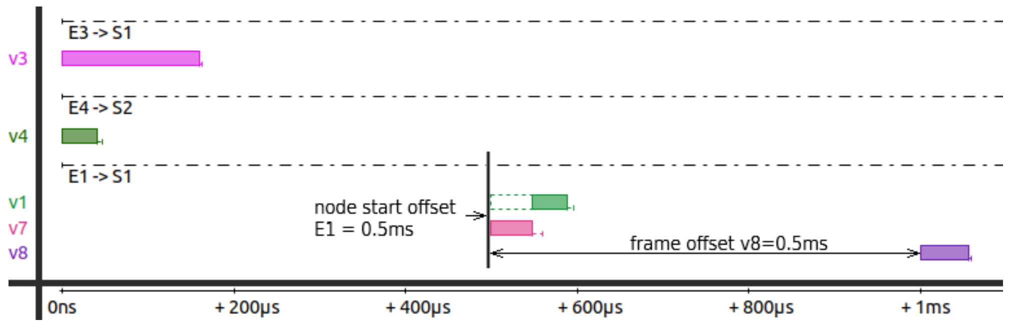


Figure 2.4: **A partial sample TSN trace showing the effect of node start offsets and frame offsets.**
The trace shows traffic of the network presented in Figure 2.2. The trace is limited to the first hops of flows v1, v3, v4, v7 and v8. Flow v3 is sent from end-node E1 to switch S3, flow v4 from E4 to S2 and flows v1, v7 and v8 from E1 to S1. All node start offsets are zero except for the node start offset for end-node E1, which is set to 0.5ms. The frame offsets for all flows are set to zero, except for the frame offset of flow v8, which is set to 0.5ms. It can be seen that the initial packets of flows v3 and v4 are instantly released at t=0ms. The initial packets for flows v1 and v7 on end-node E1 are only scheduled after the NSO(E1) of 0.5ms has elapsed. The packet of flow v7 is transmitted first, while the initial packet of v1 has to wait until v7 is done transmitting. The initial packet of flow v8 is only released at time 1.0ms, which is composed of NSO(E1)=0.5ms and FO(v8)=0.5ms.

2.2.2 Network Topology

The network topology defines which end-nodes and switches are connected together. Different physical characteristics and constraints of the system must be considered when designing a network topology. For instance, sensors and cameras must be placed in specific locations when building an autonomous car, introducing certain non-functional constraints about the required bandwidth and how they can be physically connected to the network. The aim is generally to develop topologies that minimize the cost of the overall network, thus minimizing the number of end-nodes, links and switches, while fulfilling other requirements like extensibility, safety and reliability. While the physical locations are often abstracted by models, the physical closeness to a switch can be a deciding factor in the topology design to reduce wire length and, with it, cost and weight.

Links

Links are depicted in Figure 2.2 as blue lines that connect nodes. For nodes to be able to communicate with each other, they need to be connected directly or by transitivity. All links encountered in this dissertation are considered to be full-duplex Ethernet links, meaning communication can travel in both directions simultaneously without affecting either direction. Typical transmission speeds of Ethernet links are 100 Mbit/s, 1Gbit/s and higher.

Network Bridges / Switches

The other important component to interconnect multiple end-nodes is network bridges that allow to connect different parts of a network. Different types of network bridges exist, but in this dissertation, the focus is on multi-port wired store-and-forward Ethernet switches, which are generally referred to as ”switches”. These Ethernet switches are in charge of receiving data packets on an ingress port, performing Quality-of-Service management, storing them in queues if necessary, and forwarding them accordingly to the egress port designated for that data packet. Typically, an end-node is connected to a single switch, and switches are typically connected to at least three components, which are either end-nodes or other switches. There are no restrictions to creating circular topologies, like a ring of switches connecting to various end-nodes, for instance. The switches are depicted in Figure 2.2 as blue squares and their names start with ”S” in the shown example.

Switches contain multiple ports that allow them to connect to end-nodes or other switches. Typically, ports on which data is received are referred to as ingress port, while ports on which data is sent are referred to as egress port. In the context of this work, since we are investigating full-duplex systems, every port fulfills both roles, ingress and egress. However, sometimes, ports may be referred to as ingress or egress specifically, depending on their role regarding the routing of a specific flow under investigation. In that context, an port is denoted as ingress if data packets are received by it in the routing of that flow, and denoted as egress port when it is used by flow packets to exits the switch in the forwarding direction.

2.2.3 Data Traffic

Data transmitted on a network is typically analyzed in the unit of a data frame. A data frame is a sequence of bits of a certain length that represents one piece of information bundled with a header that contains meta-information on the packet. Data frames are often also referred to as data packets. Data frames in transmission are depicted in Figures 2.3 and 2.4 as filled colorful rectangles. A data frame typically consists of one or multiple headers and a payload, which represents the data content that is supposed to be transmitted to the receiver. The header contains meta-information about the packet, such as its bit length or priority, and is used to identify, process and transmit the packet to its correct destination.

The action of a data frame passing from one network segment via a network bridge to the next segment is known as a hop. During a hop, data packets face multiple delays, including transmission, propagation, processing and queueing delays.

(Worst-Case) Delays / Traversal Times

End-to-end delays, or end-to-end latencies, describe the time a data frame takes from the moment it is scheduled for transmission at the sender node until the moment it is fully received by the destination end-node. Such an end-to-end delay is composed of the sum of local hop delays, or traversal times, that result from different processing steps that occur when traversing a network bridge. Such hop traversal times are composed of transmission, propagation, processing and queueing delays:

- The transmission delay is the time it takes for the sender to completely transmit the packet data to the link. This sender can be an end-node or a preceding switch on the route of the packet.

- The propagation delay is the time the electrical current needs to travel through the cable from the sender to the receiving switch or end-node.

- The processing delay is the time the switch needs to process the meta-information contained in the packet header and to decide how it is handled, queued and forwarded.

- Queueing delays occur when a packet must be queued and cannot be forwarded directly to the next node because the egress port is busy. How the queueing is handled depends on the specific quality-of-service mechanism that is employed, which depends on the network design and will be discussed later in this section.

The maximal time a flow packet may need to be transmitted from the sender to its receiver is referred to as the worst-case end-to-end delay, also known as worst-case traversal times (WCTT) in the literature. The term WCTT refers to the actual precise maximal end-to-end delay that can be observed for a given flow. Approximations of the WCTT are typically referred to as upper bound or lower bound, depending on the direction of the approximation.

The term ”traversal time” in the literature is often used to refer to local delays occurring, for instance, inside a switch, spanning multiple network segments or also end-to-end delays. In this work, the term is generally used synonymous with end-to-end latencies and end-to-end delays. This is done to remain consistent with the term worst-case traversal times (WCTT), which also describes end-to-end delays rather than local delay.

Data Flows

The traffic of networks investigated in this work is considered to be known at design time and is typically described by entities that precisely define its characteristics and behavior. These traffic entities are streams of data frames and are known under various names, such as ”data flow”, ”packet flow”, ”traffic flow”, ”frame flow”, ”data stream”, ”packet stream”, ”traffic stream” or ”data transmission”. Sometimes, these entities are also specifically denoted by the type of data that is transmitted, for instance, ”media stream” or ”event stream”. In packet trace illustraction, such as Figures 2.3, 2.4 and 2.5, data flows are depicted as rows, representing a sequence of transmitted data frames. For simplicity, ”data flow” and ”traffic flow” are the names of choice in this work.

Data flows define the specific traffic characteristics and properties of the data transmission, such as flow type and deadline. The real-time property of a flow is described as the deadline, which defines the maximal acceptable end-to-end delay of a data packet. Other flow characteristics include the data type and the corresponding parameterization for the assigned data flow type.

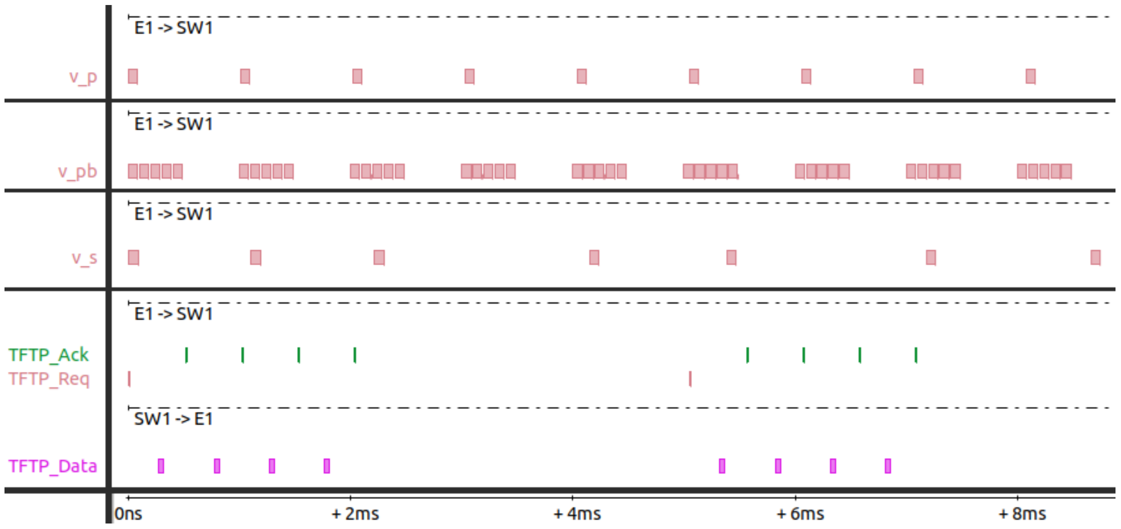


Figure 2.5: **Illustration of flow type transmission patterns.**
The illustration shows individual packet traces of example flows that are not impacted by other traffic. It can be observed that the periodic flow *v_p* and periodic burst flow *v_pb* have regular transmission patterns. In comparison, the sporadic flow *v_s* has a variable spacing between packet transmissions. For the (compound) TFTP flow consisting of *TFTP_Req*, *TFTP_Data*, and *TFTP_Ack*, it can be observed that the flow uses a conversational model. The ”Data” and ”Ack” packets are produced as a response when the ”Req” and ”Data” flows are received, respectively. In the illustrated example, the TFTP flow is configured such that a processing time of 0.2ms elapses before a response packet is produced.

Flow Types

The flow types considered in this work include periodic, periodic burst, sporadic, and protocol-based flows, as shown by the example of TFTP. These different flow types have different properties:

- **Periodic flows** send data packets of a fixed or variable size up to a set maximum. These packets are scheduled for transmission at a fixed interval known as "period". This type of flow is typically used for data that is transmitted at a regular interval, for instance, reading sensor information. In Figure 2.5, the flow *v_p* illustrates the transmission pattern of a periodic flow sending a single packet every 1ms.

- **Periodic burst flows** are flows that emit a series of packets for a combined fixed size. Bursts of packets spaced by a minimal delay are scheduled periodically. This flow type is used, for instance, to transmit video frames subdivided into multiple packets. The flow *v_pb* in Figure 2.5 illustrates the transmission pattern of a periodic burst flow sending bursts of five packets every 1ms.

- **Sporadic flows** send data packets of a variable or fixed size. Frame emissions are temporally spaced by a minimal delay. In AFDX, this minimal delay is known as bandwidth-allocation-gap (BAG) and is used to steer the maximal resources a flow can consume. A sporadic flow is represented in Figure 2.5 under the identifier *v_s* and illustrates the irregular distance between packets with a minimum distance of 1ms. An example of a sporadic flow is, for instance, external input such as opening the passenger door of a car.

- **Request-Reply Protocol flows** create a compound dialog-style communication between end-nodes. First, a request packet is sent from a sender to a receiver, which then replies once the request packet is received and processed. These types of flow differ from periodic and sporadic flows in that there is a precedence constraint between subsequent packets of a conversation round as response packets are triggered by a preceding reception.

  One such type of flow investigated in this dissertation is "Trivial File Transport Protocol" (TFTP), used to transmit data files. The protocol is initiated by a request packet and is then concluded by reply-acknowledgment rounds, that are repeated until the file is completely transferred. Figure 2.5 illustrates the transmission pattern of such a compound TFTP flow in the form of three flow components *TFTP_Req*, *TFTP_Data* and *TFTP_Ack*. It can be observed that the transmission is induced by the "Req" packet, triggered every 5ms and is answered by four "Data"-"Ack" rounds to completely submit the requested file.

Further flow types that are not considered in this dissertation exist and may be used to model more specialized behavior depending on the use case. The flow types introduced above are the most prominent and should cover most use cases.

Frame Offsets

Frame offsets (FO), are a feature that can be used to control the scheduling of frame packets between different flows, by setting an offset with respect to the clock of the sender node (local clock) or with respect to the reference clock (global clock). By setting a frame offset, the initial scheduling of a flow, and thus the start of the periodicity, is shifted by the specified time.

The effect of frame offsets (with respect to a local clock) is demonstrated in Figure 2.4, where all flows except for flow v8 have a frame offset FO(f)=0 and start sending data as soon as the sender node is ready to send data (i.e. after the NSO time has expired). Flow v8 has a frame offset of 0.5ms, and it can be observed that flow v8 starts sending data only at time t=1ms, which is composed of 0.5ms NSO of the sender end-node E1 plus the frame offset of flow v8.

Frame offsets are usually used in the context of synchronization and it usually requires specialized hardware or software to control these offsets. In this dissertation, synchronization is not considered and frame offsets are applied as a means to control the sending order of flows originating from the same sender. Consequently, the frame offsets considered in this work are always assumed to be relative to the local clock of the sender nodes.

Routing

Routing defines one or multiple paths from a flow sender node to its receivers through the network topology. An example route for a flow v1 is depicted in Figure 2.2 highlighted by the dashed red line.

Typically, there is a path from a sender to each receiver. In the case of multi-cast flows, it is generally preferred to share as much of the route to different receivers as possible to reduce the network load. The packet is only sent once by the sender node and is replicated by a switch and forwarded via different egress ports once the route splits toward different receivers.

In the case of redundancy to increase the reliability of the communication, a single flow can also have multiple paths from the sender to the same receiver and can be replicated at the endpoints of the route or along the route in switches according to the TSN 801.2CB standard, also known as ”Frame Replication and Elimination for Redundancy”.

To be able to draw conclusions about the latencies and other measures of the network, the routing of the data flows through the network needs to be known and remain constant. In the context of this work, the routing is static and defined at design time.

2.2.4 Network Characteristics

Different network characteristics can be of interest when modeling a system. Generally, these characteristics include end-to-end latencies, link loads, switch queue length, and potentially many others like packet loss, depending on the model’s level of detail. This dissertation focuses on end-to-end transmission delays, but the aggregation approach may be useful to investigate other characteristics, too. Network characteristics can be observed in a specific system state or as a statistical measure over a period of time.

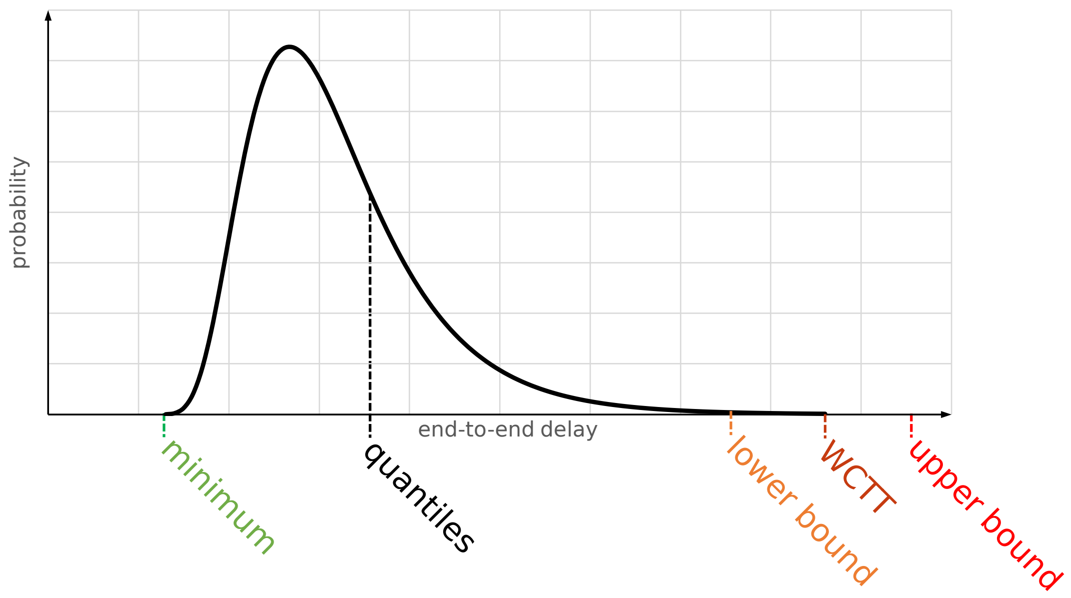


Figure 2.6: **Statistical measures of an end-to-end delay PDF.**
The illustration annotates different measures of interest for a probability density function (PDF) for end-to-end delays of a flow. The x-axis shows the magnitude of the observed end-to-end delay and the y-axis represents the probability of observing such a given magnitude. The observable end-to-end delays of a flow are always located in the interval between the minimum and maximum (WCTT) values. The annotated upper bound is a value superior to the WCTT, as typically given by analytical methods, while the lower bound is an approximation as given, for instance, by simulation. Quantiles indicate the value at which a percentage of observations in a dataset fall below the quantile threshold. For instance, we denote a quantile Q6 ($1-10^{-6}$ quantile) as the value that is exceeded by, on average, one in a million observations. The curve displayed is a log-normal distribution and does not represent real data. In a real case, the PDF of flow delays can vary significantly but the annotated statistical measures retain their properties as described.

The distribution of end-to-end delays of a flow observed over a certain time span can be represented as a probability density function. Statistical measures of interest include minimum, maximum, average, median, and different quantiles. As the exact value of the worst-case traversal time (WCTT) of a flow is unknown, it is typically investigated in terms of lower and upper bounds. Another measure to analyze end-to-end delays and other network characteristics, are quantiles. A quantile describes a value below which a certain portion of the observation falls. As rare events are of high importance in real-time networking, very large quantiles are typically considered. For instance, the quantile Q3, or $1 - 10^{-3} = 99.9\%$ quantile, describes the value which is exceeded by one in 10^3 observations. These measures are illustrated based on an example probability density function in Figure 2.6. Quantiles and PDFs are discussed in more detail in Appendix A.2 and A.3.

End-to-end Latencies

End-to-end latencies have already been briefly introduced in the delays section. Even though end-to-end latencies and traversal times are related concepts, and are often used interchangeably, they differ in certain details. End-to-end latencies are also known as end-to-end delays. The end-to-end delay describes the time it takes for a data packet to be transmitted to its designated receiver end-node after it has been scheduled for transmission by the sender. Traversal times, on the other hand, are often used to refer to local delays when analyzing latencies that occur in certain components or segments of the network. In the context of bursty traffic flows, the end-to-end latencies describe the time between the scheduling of the first packet of a burst and the reception of the final packet. For the TFTP flows in this dissertation, however, the end-to-end latencies are considered per component-flow, which means the delays of ”Req”, ”Data” and ”Ack” flows are considered individually.

The term worst-case traversal time (WCTT) refers to the maximal end-to-end latency that can occur in the given network and does not refer to local delays. In this work, the terms end-to-end latencies, end-to-end delays and traversal times will be used interchangeably. Local delays will generally not be considered unless explicitly stated.

Real-Time Property

The real-time property is defined by a deadline for the end-to-end delays of flows. Real-time systems (RTS) are categorized into hard, firm, and soft real-time, depending on their requirements as described by Kopetz in [60].

In **hard** RTS, the deadline is considered strict, and the transmission must be completed within that time frame. If a deadline is missed in hard real-time systems it may lead to catastrophic failures. An example of a hard RTS is, for instance, the brakes on a car where the actuation must take place within a guaranteed deadline after the activation.

In contrast, in a **firm** RTS, missing a deadline does not lead to catastrophic failure, but the lateness of the transmission may render the results useless or degrade their worth. An example of such a system is the measurement of the temperature in an industrial furnace. If the temperature result data arrives late, it may not be as precise or correct anymore, but it would still remain in the close range of the actual value and thus not lead to an instant critical failure of the system.

Soft RTS are designed to meet deadlines most of the time, but occasional deadline misses can be tolerated and do not lead to a direct system failure. An example of such a system could be a camera surveillance system. If a camera frame is delivered late occasionally, it may result in some stuttering of the recorded or observed video, but it does not lead to a failure of the system as long as the deadlines are met most of the time.

Schedulability in real-time systems is the property that expresses whether all flow deadlines can be met in all possible scenarios according to the system model and configuration. If this property holds true, as real-time systems is then said to be schedulable.

Quality-of-Service

Quality-of-Service mechanisms (QoS) are employed to manage traffic and avoid congestion. Many different QoS have been developed for different applications and needs. The most basic QoS that are employed are first-in-first-out (FIFO) and fixed-priority (FP/FIFO) scheduling.

As the name suggests, the FIFO mechanism forwards packets in the order of their reception. This can quickly lead to congestion as traffic of low importance can significantly delay traffic of higher importance, which typically has shorter deadlines. To address this issue, fixed priorities (FP/FIFO) were introduced as a QoS mechanism. The traffic of higher priority can overtake the lower priority traffic queued for transmission in switches. When a packet of high priority arrives at a switch, it is transmitted before the packets of lower priority that are queued in the switch but needs to wait for same or higher priority traffic if present in the queue. Traffic of equal priorities is processed according to the FIFO mechanism, which is expressed by the acronym FP/FIFO. It should be noted that in this dissertation, frame preemption is not considered as it requires specialized hardware. Frame preemption (IEEE 802.1Qbu) is the ability of a switch to interrupt the sending of a frame currently in transmission. When preemption is not supported, it leads to an effect known as ”priority inversion”, and describes the fact that a packet of lower priority can delay a packet of higher priority in a switch if that packet’s transmission started before the higher priority packet was fully received and processed by the switch.

Other QoS mechanisms that are introduced by TSN include Credit-Based Shaping (CBS) and Time-Aware Shaping (TAS). CBS functions by allocating a certain credit to each flow, which is used up during transmission. If a flow runs out of credits, other flows are prioritized, and the flow has to wait until enough credit is accumulated again. TAS is a QoS mechanism that requires node clocks across a network to be synchronized and allocates a sending time slot for each flow. In contrast to the mechanisms presented earlier, which implement rate-constrained traffic management, TAS implements time-triggered traffic.

QoS mechanisms impact how different flows interact with each other in switches and thus react to the effects of the starting conditions differently. QoS mechanisms can neutralize the effect of node start offsets to a certain extent. TAS, for instance, shifts the complexity towards generating time schedules for flows and synchronizing node clocks instead.

2.3 Analytical Methods

Analytical methods employ a mathematical model to compute certain properties of real-world systems, such as the end-to-end delays. These methods are typically preferred to assess delays in hard real-time networks since they give firm guarantees if the assumptions made by the model hold true. Important limitations of analytical methods include the possibility of extensive pessimism, limited scalability, and availability.

Many different analytical approaches to evaluate real-time systems have been developed over the years, each with its advantages, disadvantages, and use cases.

2.3.1 Holistic Method

An early approach to evaluate distributed hard real-time systems, known as holistic method, was proposed in 1994 by Tindell in [99] and extended by Tindell and Clark in [100]. A window-based analysis approach by summing local delays is developed to compute an upper bound on the worst-case response times in a preemptive environment with shared resources. The approach was later extended for non-preemptive scheduling on uniprocessor systems in [43].

In 2005, Martin and Minet applied the approach to real-time networks in [73] and showed that it generates pessimistic upper bounds. In 2012, Gutiérrez, Palencia and Harbour applied the approach to AFDX networks in [46].

2.3.2 Model Checking

Model checking [32] typically uses a timed automaton to model and verify real-time systems [63]. A model checker generally considers all valid automaton states and can thus compute the exact worst-case end-to-end delays. However, this precision typically comes at a high computational cost and does not scale to practically relevant use cases as the complexity explodes for larger network configurations, as discussed in [31], for instance. Model checking cannot only be used to compute exact worst-case traversal times but also allows verification of various other properties relevant to Ethernet networking, such as the correctness of a protocol.

Many model checkers are available that allow to model and verify real-time systems, such as UPPAAL [20], Promela/SPIN [49], [101] and others. For instance, in 2010 Adnan et al. have developed a UPPAAL-based model to compute AFDX worst-case delays that uses a port-by-port analysis to reduce the search space in [11]. Model checking in the domain of Ethernet-based networks was applied in a number of use cases, including computation of worst-case delays [17], [31], [10], and properties such as reliability, correctness and others [14], [52], [44], [70].

Despite efforts to reduce the search space for model checking and thus improve scalability, for instance, in [13], [12], [64], model checking remains non-practical for large industrial use cases.

2.3.3 Network Calculus

Network Calculus (NC), originally developed by Cruz in [33] and [34] served as a basis for analytical approaches to evaluate network performance described by Chang in [28] and Le Boudec in [65], [66].

Network Calculus is a mathematical framework built on $(min, +)$ algebra that models all network components as arrival and service curves. Using these curves, the incoming and outgoing traffic can be modeled to analyze the behavior and performance of computer networks. Specifically, the framework aims to analyze properties such as flow delays, data throughput, and link capacity.

Network Calculus has remained one of the most popular methods to evaluate network delays as it gives firm guarantees on the WCTT and scales rather well compared to many other formal methods. In fact, Network Calculus is used for certification as described in [71]. However, due to the over-approximation of the traffic function by the service and arrival curves, Network Calculus is generally pessimistic to an unknown degree. The pessimism of Network Calculus has been investigated in various works, for instance, by Navet et al. in [80].

In [31] Charara, Scharbarg, Ermont and Fraboul compare simulation, network calculus and model checking for bounding end-to-end delays in the case of an AFDX network.

2.3.4 Trajectory Approach

The trajectory approach (TA) was developed by Martin and Minet to analyze delays in networks. It was proposed in 2005 as an improvement over the holistic approach in [73] and further developed for first-in-first-out (FIFO) [74], fixed priorities with FIFO (FP/FIFO) [75] and fixed priorities with earliest deadline first (FP/EDF) [76]. The approach is based on the principle of determining the impact of other packets that can delay the packet of interest along each node in its trajectory from sender to receiver. Iteratively and starting at the last node, packets are then placed according to the scheduling rules to maximize the busy period and thus the encountered delays across all nodes.

An improvement to the trajectory approach to compute end-to-end delays in AFDX networks together with a heuristic to generate unfavorable optimistic scenarios, also known as unfavorable scenario analysis, was developed by Bauer, Scharbarg and Fraboul for FIFO traffic in [18] and for FP/FIFO traffic in [19]. This unfavorable scenario approach has the advantage that it allows the generation of an actual realistic packet trace. These unfavorable scenarios stringently follow the rules of the system model and traffic management mechanisms, and could thus be observed in simulation. The approach yields tight lower bounds on the exact worst-case end-to-end delays. The effectiveness of the approach to approximate WCTT was investigated by Boyer et al. in 2012 in [24].

The pessimism of the trajectory approach is analyzed by Li, Scharbarg and Fraboul in 2011 in [68] and also in 2012 in [78] by Medlej, Martin and Cottin for FIFO scheduling. Medlej further aims to improve the pessimism and scalability of the approach in [77].

On the other hand, the trajectory approach was found to be optimistic in some corner cases. These corner cases for FIFO scheduled traffic are discussed in [57] and [58] by Kemayo, Ridouard, Bauer and Richard in 2013. Li, Cros and George propose a solution to this optimism in [67] without formal proof.

In 2019, Tang, Li, Lu and Xiong proposed a revised trajectory approach in [94] to reduce pessimism while also eliminating the optimistic corner cases.

2.3.5 Real-Time Calculus

Real-Time Calculus (RTC), proposed in 2000 by Thiele, Chakraborty and Naedele in [97], is based on the principles of Network Calculus [28] combined with schedulability concepts. They further derive a polynomial algorithm for feasibility analysis and optimal priority assignments.
In 2006 Wandeler, Thiele, Verhoef and Lieverse proposed in [105] an approach based on RTC using a high abstraction level with the aim to enable early design space exploration in the development of such systems.

A probabilistic real-time calculus is proposed in [89] and [90] by Santinelli and Cucu-Grosjean with the aim to analyze systems that are non-deterministic but defined in terms of random variables.

In 2019, Zhang, Liu, Shi, Huang and Zhao proposed in [110] a framework to compute feasibility of TSN networks using RTC.

2.3.6 Compositional Performance Analysis

In 2012, Diemer, Rox and Ernst proposed in [37] a Compositional Performance Analysis (CPA) based on RTC, specialized to analyze AVB traffic. In [38], Diemer, Thiele and Ernst generalize the approach to AVB and strict-priority Ethernet traffic. CPA was extended to strict-priority scheduling in 2015 by Thiele, Axer and Ernst in [95]. Efforts to exploit flow correlations to improve the tightness of bounds generated by CPA were made in [96] in 2014 by Thiele, Axer, Ernst and Seyler. A Python implementation for the CPA approach is provided in [36] by Diemer, Axer and Ernst.

CPA performs the analysis by first transforming the system into a timing analysis model via model transformation. The model transformation converts the network into a CPA model that consists of a directed graph in which nodes represent tasks and edges describe their dependencies. Event arrival curves are then used to describe the timing of task activations. Event models are then derived and propagated among dependent tasks until a fixed point is reached.

2.3.7 Forward Analysis

Forward end-to-end delay Analysis (FA) was originally proposed in [59] by Kemayo, Ridouard, Bauer and Richard for AFDX networks with only FIFO traffic. It was later improved and formally proven in 2017 by Benammar, Ridouard, Bauer and Richard in [23]. The first variant was aimed at FIFO scheduling only, which was extended to FP/FIFO scheduling in [22]. The FA approach is optimized using a flow serialization approach that has proven effective in other methods. In 2018, the approach was extended to support credit-based shapers for AVB networks in [21].

Generally, FA is performed by iteratively analyzing the worst-case scenario for each node along the flow route from sender to receiver. For each of these nodes, the maximum backlog is computed to determine the local worst-case waiting time. This delay is propagated until the receiver end-node to determine the worst-case delay and is repeated for each flow in the network. In 2020, Xu and Yang proposed an optimization to FA considering the serialization effect of packets in [109] and evaluate the performance gain on example and industrial AFDX Networks.

2.4 Network Simulation

Network simulation aims to investigate the communication behavior based on a model of a real system. This is done by modeling the state of the underlying components, such as links, switches and end-nodes. The traffic that constitutes the communication is modeled as data packets that are generated based on abstract definitions of the communication needs or even by simulating or running the actual end-node software if available.

Simulation is necessary in developing and designing critical embedded real-time systems for industrial applications. It allows for quick approximation and assessment of different properties of the architecture, starting at the early stages of the development process when not even all specifications may be precisely defined. As the architecture evolves, the simulation can be refined with more precise constraints and can even be integrated with partial prototype testbeds. It allows for the quick assessment of the effects of different topologies, mechanisms, protocols, hardware, and software options. Many different properties of the system can be analysed this way, from link loads and memory requirements to end-to-end communication delays. This enables the designer to make informed choices to develop an architecture that is suitable to the requirements and constraints of the system.

In contrast to analytical methods, simulation produces lower bounds on the WCTT, as observing the absolute worst-case is very unlikely but technically possible if the model is correct. End-to-end delays observed in simulations are subject to probabilistic effects. Scenarios leading to very high end-to-end delays are typically rare and can require much simulation effort.

Another advantage of network simulation is that it enables the computation of quantiles from the observed distributions of the different network properties. This is particularly interesting when developing fault-tolerant systems that can recover from a certain frequency of errors in transmission, for instance.

Simulation models are also generally more flexible to be adjusted to new technologies, as extensions can be implemented according to specifications. Developing and optimizing analytical methods, on the other hand, is usually more complicated and cost-intensive as it requires expert knowledge in the domain of the analytical approach, like Network Calculus, for instance. Simulation further gains importance as industries strive towards implementing ”Digital Twins” [61], [29], [84], enabling a continuous evolution and validation of systems.

The remainder of this section will discuss Discrete-Event Simulation, the predominant method to implement network simulators, different open-source and commercial simulation softwares, and will give an overview on simulation parameters and metrics.

2.4.1 Discrete-Event Simulation

Discrete-Event Simulation (DES) is the predominant method to simulate network behavior [83]. It allows the simulation to be limited to only certain states of the system, during which relevant events that change the state of the simulated system occur. These system states will be referred to as ”simulation states”.

For instance, instead of modeling every instant of the continuous transmission of every bit of a packet as electrical current, Discrete-Event Simulation abstracts this as an event of transmitting a whole packet over a link and computes certain properties of that event and the effects it has on the state of the involved components of the network. Properties of such a transmission event could be, for instance, the time at which the packet starts to be transmitted, the time at which it is fully received by the switch, and the change in the switch is recorded where the packet is now located after transmission. The processing of this event then creates a new one, which would constitute the forwarding of the packet from the switch as a future step. After processing one event, the time of the simulator is then advanced to when the next event is due. This abstraction allows to efficiently simulate many packet flows with differing properties traversing complex network architectures.

Alternatives to Discrete-Event Simulation are Discrete-Time Simulation and continuous simulation. Discrete-Time Simulation progresses the time of the system in fixed time increments rather than skipping to the next event. Consequently, it is less efficient than Discrete-Event Simulation while maintaining the same general working principle. Another down-side of discrete time simulation is that events in the system may not perfectly align with the granularity of the chosen time-step, which would thus add additional complexity or imprecision.

Continuous simulation, on the other hand, constitutes a significantly different paradigm that relies on differential equations to model system behavior and is typically used to model analogous and physical systems, that can continuously change their state. In [16] discrete and continuous simulation approaches are defined and compared in more detail.

2.4.2 Parallel Discrete-Event Simulation

Discrete-Event Simulation is typically sequential and can not be easily parallelized as simulation states depend on their preceding states. This is an important limitation as today's computing infrastructures rely on scaling via parallel computing resources such as multi-core CPUs, GPUs and HPCs. As a consequence, research was conducted to overcome this limitation in many fields where Discrete-Event Simulation is applied. Some parallelization challenges that are faced across various research areas, including network simulation, are discussed in [83].

The general goal of Parallel Discrete-Event Simulation (PDES) is to achieve results that are equivalent, or ideally identical, to the results achieved from sequential simulations. To that end, two main types of parallelization are considered: spatial and temporal decomposition. Hybrid approaches that aim to combine both have also been researched.

Spatial parallelization is the most prominent approach and is typically concerned with decomposing the model into multiple local elements that are then simulated in temporal order and are then aligned on the decomposition boundaries to lead to overall convergence.

Temporal decomposition aims at parallelizing the simulation in the time domain, which is challenging due to the dependency of simulation states on their predecessors. Approaches to achieve temporal parallelization include synchronizing computing entities, predicting and rollback of simulation states, or relaxation of precision requirements. Many different techniques to achieve either variant have been developed and are discussed widely in [81].

2.4.3 Simulation Time vs. Simulation Duration

Two types of time will be considered when discussing different reference systems, namely the simulation time and the simulation duration. Both terms can be used to refer to different properties of a simulation, and are defined and explained below.

Simulation Time

The ”simulation time” or ”simulated time” describes the time that passes inside of the simulated system, with respect to its reference clock. Another way to think of it is virtual time that passes inside of a closed system. The simulation time defines the global reference clock introduced earlier and acts as the time reference for the simulation states. As discussed in the section about Discrete-Event Simulation, this time can be fast-forwarded to the next event during simulation. It is the time that is attached to simulation events or simulation traces inside the simulator.

Simulation Duration

The term ”simulation duration”, on the other hand, refers to the time that passes outside of the simulation, respectively, in the physical world. It is the time needed to execute the simulation and generate the simulation results. It refers to the time that passes for the user waiting for the simulation to be performed by the host machine. It is the time that passes outside of the simulation. This represents the ”real” time that passes in the physical world, oftentimes referred to as wall-clock time in the literature.

2.4.4 Speedup Factor Variants

Three different types of speedup factors will be differentiated in this dissertation. Those three factors will be referred to as parallelization speedup factor $Speedup_{par}$, aggregation speedup factor $Speedup_{agg}$, and simulation speedup factor $Speedup_{sim}$. The naming of the different speedups indicates their respective sources. The different speedup factor variants will be defined next and are independent of each other, which means they can be impacted separately and combined to increase the overall speedup achieved by the aggregation approach.

Parallelization Speedup Factor

The parallelization speedup factor results from the parallelization of a portion of a task or program and as given by Amdahl's Law [48] and defined as

$$Speedup_{par} = \frac{1}{(1-f)+\frac{f}{n}}$$

where f is the parallelized fraction of the task, and n is the number of available processor cores or, more precisely, the parallelization factor.

Even though not explicitly researched in this dissertation, this factor is important as the aggregation approach unlocks additional parallelization potential as discussed in Section 3.4.3 and must not be confused with aggregation or simulation speedup.

Aggregation Speedup Factor

As will be observed in Chapter 4 and Chapter 5, the aggregation approach allows the production of equal overall traversal time bounds to long simulations with a significantly reduced total aggregated simulation time in many cases. That is, fewer short simulations need to be run amounting to a lower total aggregated simulation time compared to long simulation for achieving equal bounds. This can be represented as an aggregation speedup factor

$$Speedup_{agg} = \frac{t_{long}}{t_{agg}}$$

where t_{long} represents the simulation time budget of the long simulation and t_{agg} represents the aggregated simulation budget for which the bounds of the long simulation are matched or exceeded by the bounds generated via aggregated short simulations. In Chapter 4 and Chapter 5 these delay bounds will be represented via the AMTT metric introduced later in Section 4.2.4.

It is important to note that the aggregated simulation budget is expressed as the sum of the simulation times of the aggregated simulations and is thus considered non-parallelized. Therefore this speedup factor exists independently of the parallelization speedup factor and simulation speedup factor, which will be introduced next. It is important to differentiate the aggregation speedup and the simulation speedup as they are impacted by different factors. For instance, the simulation speedup is impacted by the host machine's computation speed, which does not impact the aggregation speedup. On the other hand, both are impacted indirectly by the chosen simulation time.

Simulation Speedup Factor

The simulation speedup factor defines the relative factor of how the simulation time evolves compared to the simulation duration. As a reminder, the simulation time is the time that elapses inside of the simulated system, while the simulation duration is the time it takes to run the simulation in the physical world. It is directly impacted by the computation speed of the host machine as processing the simulation faster results in a reduction of the simulation duration. The simulation speedup factor *SP* can be defined as

$$Speedup_{sim} := \frac{t_{sim}}{t_{dur}}$$

where t_{sim} is the simulation time and t_{dur}. This factor is expressed as a non-parallelized measure, so the simulation time and duration are both expressed in relation to a single simulation or an aggregation of simulations run sequentially.

If a speedup factor is smaller than one, it means the simulation takes longer to be simulated than it would take to run the modeled system. Depending on the complexity of the system, the speedup factor may also be larger than one, which means we can simulate multiple hours of functioning of the simulated system in just one hour of computation time.

The speedup factor plays an important role when applying the findings of this dissertation to practical cases, as reducing the simulation time can negatively impact the speedup factor and thus reduce the efficiency of the simulation aggregation approach.

Another important aspect of the speedup factor is that it represents an average speedup across the whole simulation duration. The time it takes to simulate a certain portion of the simulation trace depends on the number of simulation events that occur during that portion. The simulation time is thus not linearly correlated with the simulation duration. It also means that the simulation duration can vary for simulations of equal simulation time of the same system if parameters that influence the traffic are changed, such as, for instance, the random seed.

2.4.5 Simulation Traces and Statistical Measures

The data that is produced during a simulation can be used by the simulator to compute statistical measures as introduced earlier, or it can be stored as simulation traces containing all sorts of information depending on the specific simulation model and the simulation parameters. Such simulation traces regarding the local traversal times are depicted, for instance, in Figures 2.3, 2.4, 2.5 and 4.2.

When storing the simulation traces, large amounts of data are collected quickly, which can significantly slow down the simulation. For this reason, some simulators allow to configure what specific information is stored. The simulation software used in this dissertation, RTaW-Pegase, allows setting whether all traces should be stored, only traces that represent the worst observed scenario or no traces at all, in which case only statistical measures like quantiles, minimum, and maximum are stored.

2.4.6 Simulation Software

Different software products are available to perform network simulations. Each of them has its specific advantages and applications. In this dissertation, all simulation experiments have been performed using RTaW-Pegase [25], a commercial toolbox designed to evaluate real-time networking systems. It provides an optimized simulation engine specifically developed to run simulations for bus-based and Ethernet-based systems. The toolbox offers state-of-the-art support for most TSN standards and bus systems, and provides capabilities to run analytical methods based on Network Calculus and unfavorable scenario analysis for sporadic FIFO traffic.

All experiments around the simulation aggregation paradigm presented in this dissertation were implemented as a combination of Python and Java programs. Python was used to implement the experiment and optimization logic, and Java to run the simulations using the Java API library provided by RTaW-Pegase to run simulations and schedulability analysis.

Besides RTaW-Pegase [25], network simulators that are used in real-time networking research are OMNet++ [104], OPNet [30] and ns-3 [86]. In [107], Wehrle, Günes and Gross provided an in-depth introduction to general network simulation and described the implementation fundamentals of two popular network simulators, OMNet++ and ns-3.
A more recent comparative survey from 2020 on network simulators, with a focus on TSN,

is provided by Xie, Li and Gao in [108].

OMNet++ is a core framework for network simulation frequently used in the literature to simulate and analyze Ethernet-based networks. It is implemented in C++, uses its own domain-specific language called NED to compose simulation models and offers various tools to interact with the simulation. It uses a GPL-like license that makes it free to use for non-commercial purposes. To support Ethernet networks in OMNet++, the INET [6] framework is required.

To support TSN, a simulation framework called TSimNet was developed by Heise, Geyer and Obermaisser in [47] in 2016. It focuses on non-synchronization-based elements of TSN and is built on top of the INET framework. In 2018, Jiang et al. proposed a framework [53] to support the IEEE TSN standards 802.1Qbv (time-aware traffic scheduling) and 802.1AS (time synchronization). It is not openly available and builds on top of the Core4INET [1] framework (which did not support TSN at the time).

Yet another TSN framework, called NeSTiNg, was developed by Falk et al. in [40] in 2019. It is based on the INET framework and aims to additionally support the time-based features of TSN in comparison to TSimNet [47]. The INET framework in version 4.4.0 [3] released in May 2022 integrates TSN features into Core4INET.

OPNET [4], now known as Riverbed [5], is written in C and C++ and was originally developed for military needs. It is a commercial network simulator applicable to many use cases. In 2018, Pahlevan and Obermaisser proposed a framework based on OPNET to support TSN in [82].

The simulation software ns-3 [86] is open source and written in C++. It is less popular in the real-time Ethernet research domain. In 2020, Credit-Based Shaping (CBS) for ns-3 was implemented by Krummacker and Wendling in [61] as a first step to support TSN.

In 2020, Campanile et al. provided a systematic literature review [26] of papers referring to the ns-3 simulator to investigate its adaptation in the scientific domain. They conclude the high flexibility and success of the simulator. However, it appears that the prominence has not yet carried over for TSN research.

It can be concluded that OMNet++ is currently the most popular open-source tool [15] for the simulation in TSN network research and the only openly available alternative to commercial tools.

Chapter 3
Aggregation Approach using Random Search on NSOs

This chapter will introduce the general approach of aggregating many simulation results to evaluate worst-case traversal times. Random search via uniform sampling is applied throughout this chapter to explore the search space of specific starting conditions to investigate the potential of simulation aggregation. The effectiveness of this simple approach is evaluated based on five diverse network configurations derived from relevant industrial use cases.

First, an overview of the applications and goals targeted by the approach will be given as a reminder. Then, the intuition behind it will be explored and demonstrated by observing simulation result packet traces. The next step will be to describe and analyze the role of the different free simulation parameters that can be controlled. These free parameters include simulation time, node start offsets, flow scheduling order, frame offsets, clock drifts, and random seeds. Once we have gained an understanding of the involved parameters, the architecture of the aggregation approach will be described. After defining the overall architecture of the approach, the experiment design that was employed to answer the questions is described. Finally, the results are discussed and conclusions are drawn.

Some of the results shown in this chapter were presented in a related publication [56]. The content of this chapter is built around that work but gives extended context for an improved understanding, which would have exceeded the publication's scope. The results taken from this publication will be indicated accordingly.

3.1 Motivation and Context

The original motivation for developing the approach was to increase the flexibility of using simulation results in the context of machine learning and to increase the potential of exploring node start offsets close to the synchronized case. During the development of the thesis, the exposed potential of the approach enabled further interesting applications and research avenues beyond this original aim.

The possible applications of the aggregation approach are many-fold and include:

- **Tighter approximation of worst-case traversal times via simulation**
  Approximating WCTT via simulation is an increasingly relevant problem in practical applications. With the growing complexity of cyber-physical real-time systems, practitioners face challenges of increasing magnitude in evaluating the correctness and safety of such systems. The simulation aggregation approach, combined with optimization approaches presented in the following chapters, achieves tighter lower bounds on the WCTT than the traditional simulation approach, given equal resources.

- **Better evaluation of the pessimism of analytical approaches**
  Analytical approaches such as, for instance, schedulability analysis based on Network Calculus are often known to be pessimistic. However, in the general case, it is not precisely known how pessimistic they are with respect to the actual worst-case traversal times. Simulation provides an upper bound on the pessimism of these approaches. Consequently, achieving tighter lower bounds on the WCTT via simulation simultaneously enables a better evaluation of the pessimism of analytical approaches.

- **Improved scalability via massive parallelization**
  The aggregation approach is highly parallelizable, representing a type of problem known as ”embarrassingly parallel”. Individual short simulation instances are independent of each other and can be run in parallel to achieve a high speedup by making use of highly parallel computing infrastructures, such as HPCs. This significant speedup potential not only allows for improved scalability but also enables new types of interactive design systems. Achieving a shorter evaluation cycle allows the system designer to consider more candidate solutions and could enable a semi-automated design approach based on Artificial Intelligence (AI). Such an AI-driven system could, for instance, generate and suggest variations of candidate solutions that a human designer would typically not consider. These variations could then be evaluated quickly in parallel using the aggregation approach and provide preliminary information to the human designer, creating a more interactive iterative design loop.

- **Optimization and learning methods**
  Running many short simulations instead of long ones allows for additional flexibility while exploring the simulation state space. It means that optimization algorithms can be applied to optimize the specific starting conditions to increase the probability of observing higher traversal times, leading to significant improvements. This will be investigated further in Chapter 5, dedicated to optimizing the starting conditions for evaluating the WCTT. The investigated starting conditions include node start offsets and flow scheduling order.

Contributions

The contributions of this chapter are:

- An introduction of the general architecture of the simulation aggregation paradigm.
- The definition of applications and goals.
- An overview of the closely related work on simulation aggregation.
- A definition of the system model used throughout this dissertation.
- A discussion of the relevant simulation parameters and their effects on the observed end-to-end delays.
- An empirical analysis of the potential of aggregation of short simulations, comparing synchronized to uniformly sampled NSO.

The results of the empirical analysis will be used to answer the first three research questions, which aim to investigate the potential of simulation aggregation with synchronized and varying NSO, respectively.

3.1.1 Related Prior Work

This section will discuss fundamental prior work directly related to this chapter and partially represents a brief recap of the broader related work extensively presented in Chapter 2.

Approximation of Worst-Case Response Times via Simulation

The work by Samii et al. on the approximation of process worst-case response times (WCRT) in bus systems in [88] is closely related to the approach developed in this dissertation. Their work aims to efficiently approximate the WCRT of specific tasks in real-time systems. They propose a method to prune the search space and combine it with various exploration strategies based on genetic algorithms. The different optimization approaches are then evaluated on applications distributed over CAN and FlexRay networks. The proposed approach differs from the work developed in this thesis as it targets a fundamentally different system model. In addition, the approach proposed by Samii et al. relies heavily on expert knowledge, which limits the applicability to specific scenarios, and relies on different methods such as pruning of the search space. On the other hand, the paradigm proposed in this thesis avoids expert knowledge where possible and is thus applicable to a large variety of use cases.

Parallelization of Network Simulation

As discussed in Chapter 2, the literature differentiates between two main types of parallelization for networking simulation: spatial and temporal parallelization. Some works combine those two approaches to form hybrid parallelization solutions.

Spatial parallelization typically subdivides the network to perform simulations on parts of the network in parallel. The results of these partial simulations are then aggregated to approximate general simulation results for the whole system. Riley et al. [87] proposed a generic framework for spatial parallelization, which was integrated into the ”ns” open-source simulation software package [27].

Temporal parallelization. An alternative approach is known as temporal parallelization. It aims at subdividing the simulation in the temporal domain rather than the spatial domain. This is done by running the simulation parts starting from certain temporally spaced simulation states. Typically, the current simulation state in discrete event simulation depends on the previous state, making precise temporal parallelization hard to achieve.

To solve this problem Wang et al. [106] describe the starting conditions for these temporally spaced simulation states by approximation. They show that this approximation generally produces results that are different from those obtained with a single long simulation.

To a certain extent, the aggregation approach proposed in this dissertation could be categorized as a temporal parallelization, as the effect of modifying NSOs and other starting conditions for short simulations is approximately similar to skipping to a simulation state that is temporally spaced by an unknown amount.

Hybrid parallelization. Gupta et al. [45] propose a hybrid approach that shows certain commonalities to the work developed in this thesis. The work describes a combination of spatial and temporal parallelization methods, aiming to process very large networks using heavily parallel infrastructure.

Pessimism in Analytical Methods

Network calculus is a well-researched analysis method for verifying performance guarantees in real-time networks, as introduced in Chapter 2.2. It is known to be generally pessimistic, but the extent of pessimism is not known precisely. The pessimism may differ depending on the concrete implementation and optimizations of the analysis and the characteristics of the network configuration of interest. Many works that analyze and extend the effectiveness and applications of network calculus have been derived to this day.

In [80], Navet et al. explain the limitations of worst-case analysis based on Network Calculus for complex industrial applications and determine an upper bound on the pessimism. They further present simulation experiments that suggest the efficiency of exploring clock drifts and node offsets to maximize the observed WCTT, serving as one of the motivational foundations of the work developed in this dissertation.

Charara et al. [31] propose an analysis based on network calculus and evaluate its pessimism in the context of an AFDX network. They observe significant imprecisions on the evaluations of the pessimism of their method, a task that this dissertation aims to improve.

More recently, in 2022, a library was developed by Zippo and Stea in [111], with the aim to parallelize network calculus. It demonstrates the continued importance of the approach and the relevance of methods that can exploit parallelism.

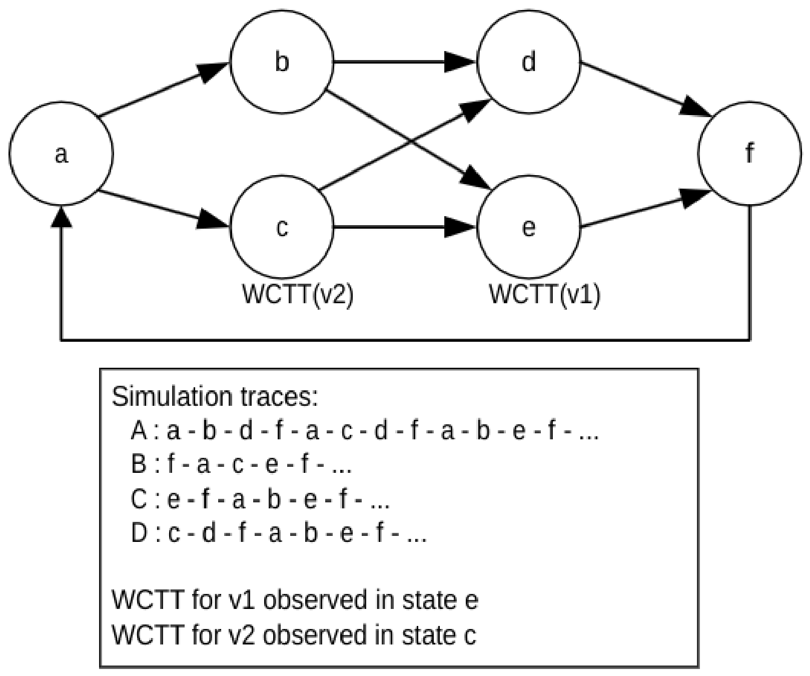
An alternative type of worst-case analysis is proposed by Bauer, Scharbarg and Fraboul in a 2010 study [18] based on the trajectory of data flows. It aims to generate concrete transmission scenarios that yield high latencies. The approach yields tight lower bounds on the actual WCTT, in contrast to network calculus which yields upper bounds. The approach yields promising results with a high quality but is not widely applicable due to the strong limitation of requiring sporadic traffic that uses FIFO. The same authors' later publication [19] extends the approach to work with fixed priorities (FP/FIFO) but is still limited to sporadic flows only. The analysis is not straightforward to implement, and it requires a significant amount of expert knowledge to adjust it to other applications.

3.1.2 Problem Formulation

The general problem is observing the WCTT for every flow via simulation. However, it is unlikely to observe the actual WCTT of a flow in simulation. Thus, to be precise, the goal pursued in this dissertation is to maximize the observed end-to-end delays for every flow and every receiver end-node. Subsequently, the event of an end-node receiving a flow transmission will be referred to as a flow reception. Since our system model allows for multi-cast flows, each flow can be received by multiple end-nodes and thus travel along differing routes. Therefore, the maximal delays for different flow receptions of the same flow are likely to be observed at different time instants and for different starting conditions.

Consequently, a solution to this problem can be represented as a set of simulation states in which the highest end-to-end delays were observed for a packet of a flow reception. These simulation states will be formally introduced later in this chapter and can be considered to be the state of the system after an event has occurred in the context of Discrete-Event Simulation as introduced in Chapter 2.4.1. To illustrate the problem from a simulation state perspective, a dummy example of such a state space is depicted in Figure 3.1, together with different possible simulation traces on that state space. It should be noted that this example only serves for illustration purposes and does not correspond to an actual simulation system. The simulation space of real applications is significantly more complex and can comprise infinitely many simulation states. However, as long as the WCTT are finite, a subset of simulation states exists at which all WCTT can be observed.

When applying the traditional approach of running long simulations to evaluate the WCTT, we rely on a property known as ergodicity. In a simplified manner, the ergodicity property can be understood as the ability of simulation to visit every simulation state when run for long enough, independently of starting conditions like node start offsets and clock drifts.



Simulation traces:
A : a - b - d - f - a - c - d - f - a - b - e - f - ...
B : f - a - c - e - f - ...
C : e - f - a - b - e - f - ...
D : c - d - f - a - b - e - f - ...

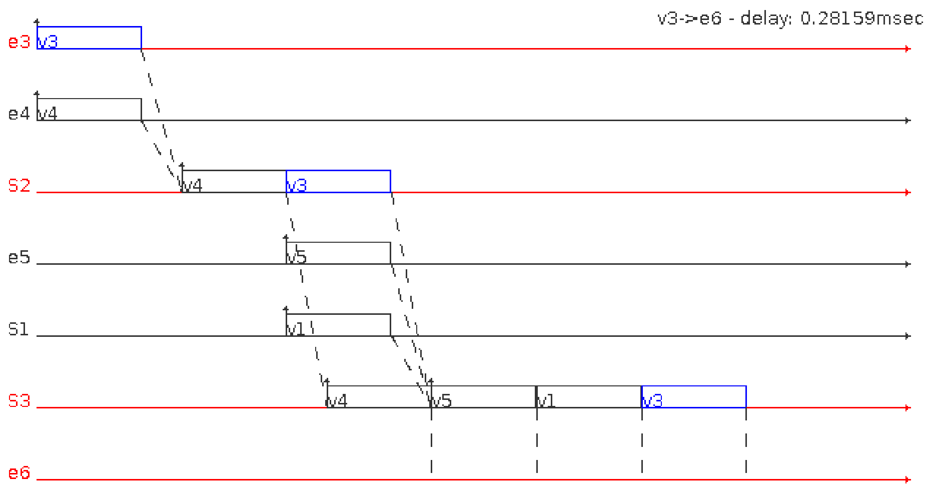
WCTT for v1 observed in state e
WCTT for v2 observed in state c

Figure 3.1: **A dummy simulation state space example.**
Every circle corresponds to one simulation state where an end-to-end latency for a flow is observed. Probabilistic transitions are displayed as multiple outgoing edges. In states ”e” and ”c”, the WCTT for flows v1 and v2 are observed, respectively. Simulation traces A-D represent possible simulations with different starting conditions. When considering the trace A, both WCTTs are only discovered after 11 simulation steps, while for trace B both WCTTs are observed after 4 steps already. If traces C and D are aggregated, both WCTT can be observed already after only two simulation steps.

Expressing the problem in terms of simulation states is not very practical as simulation states are typically not directly exposed by the simulator but rather the results (such as observed end-to-end delays) over a series of simulation states traversed during the simulation time. The problem to be solved is then generating multiple simulation scenarios or traces that, given a limited simulation time, in aggregation, maximize the observed end-to-end delays for every flow reception. A metric to evaluate the quality of an aggregation of a set of simulations will be defined in Section 4.2.4.

Adjusting the starting conditions of the simulation allows for an impact on the order in which simulation states are visited. For instance, the traditional approach of running long simulations relies on synchronized node start offsets to increase the likelihood of visiting simulation states that yield high end-to-end latencies early.

a)



b)

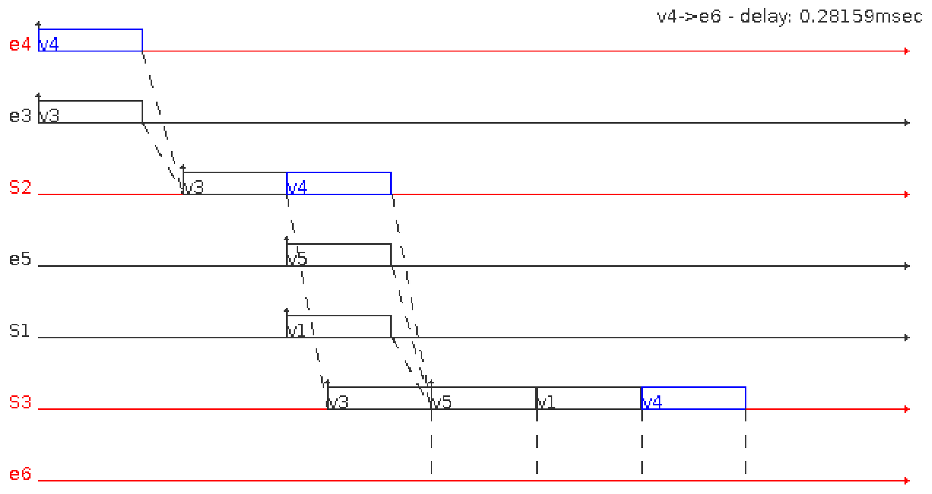


Figure 3.2: **Worst case flow traces highlighting the competitive nature of flows.** The two worst-case flow traces for flows v3 (a) and v4 (b) show mostly identical properties, except for packets v3 and v4 being ordered differently in switch S2. This example shows that no single set of starting conditions can generate the WCTT for all flows simultaneously in general. Reordering packets in switch S2 has a significant impact on the observed WCTT. This reordering can be achieved by modifying the node start offsets of nodes e3 and e4, such that one is strictly smaller than the other, allowing to steer which one arrives first at switch S2.

In practice, a solution thus consists of a set of starting conditions such that the resulting simulation scenarios would yield maximized observations of end-to-end delays. A starting condition in this context consists of a set of free parameters for the simulator that fully define the simulation behavior and thus allow the reproduction of the exact simulation results based on these parameters. These free simulation parameters are introduced and described in more detail later in this chapter and typically include a simulation seed to achieve reproducibility of the probabilistic elements during simulation.

In general, finding a single set of starting conditions that simultaneously maximizes the observed end-to-end latencies for all flows is usually impossible in the context of limited simulation time. As a consequence, mathematically speaking, a solution in the context of simulation aggregation thus consists of a subset of the set of all possible sets of starting conditions. To illustrate the necessity for a solution to consist of multiple such sets in general, Figure 3.2 shows two worst-case packet traces based on a very simple network, each using a different set of starting conditions. The packets of flows v3 and v4 in switch S2 must be reordered to generate their respective WCTT and cannot occur at the same time. As flows v3 and v4 have the same priority, this reordering can be achieved by two different means in this example. One possibility is to rely on randomness integrated into the simulator, which is steered by the random seed. The other possibility is to choose node start offsets for end-nodes e3 and e4 such that either the packet of flow v3 or flow v4 arrives marginally earlier, generating the worst-case scenario for flow v4 or v3, respectively. If flows v3 and v4 would originate from the same end-node, adjusting the node start offsets would not allow observing both scenarios, as both would be impacted by the same offset and instead, the random seed would be one means to adjust the reordering. Yet another possibility to control the flow scheduling order more deterministically is via frame offsets, which will be discussed in more detail later.

Further, there may not be a unique simulation state that maximizes the observed end-to-end delays for a specific flow and receiver end-node but rather a set of equivalent solutions, which are said to be non-dominated. This can easily be seen when considering a node from which no flow originates that interferes, directly or by transitivity, with the flow of interest. Independently of the node start offset of this node, the observed delays of the flow of interest will not change, thus rendering the solution non-unique.

3.1.3 The Traditional Approach to Evaluate WCTT via Simulation

The previously known most effective approach to approximate WCTT via simulation involves running one or multiple very long simulations starting from synchronized node start offsets (NSO) and randomized node clock drifts. In the following, this approach will be referred to as ”baseline solution”, ”traditional approach”, or ”synchronized case”. The labeling ”synchronized case” stems from the fact that the node start offsets are equal and typically set to zero. It means that all end-nodes start sending data at the exact same instant, which is very unlikely to be observed in the real world.

The idea behind this approach is that it creates scenarios, known as ”critical instants”, that yield high interferences of flows in switches and thus generate high traversal times. In fact, when all nodes start sending data at the same time, it is probable that at the first switch connected to the node, many packets arrive at the same time, which then need to be queued and cause increased delays of the packets placed lower in the queue. For example, looking at Figure 3.2 a), packet v3 arrives at switch S3 slightly after packets v5 and v1 and is thus delayed by the time it takes for those two packets to be fully transmitted before packet v3 can be forwarded.

The randomized node clock drifts help to break the hyper-period of the system and thus allow exploring the simulation search space as the simulation progresses causing the relative sending instants across the different nodes to slowly drift apart, as demonstrated in Chapter 2 Figure 2.3.

3.2 Research Questions

For this chapter, which aims to evaluate the effectiveness of the aggregation approach using random search on a fixed node start offset range, a total of three research questions were designed. They have been addressed in the publication [56], which served as a foundation for this chapter. The research questions are as follows:

RQ1. Does simple aggregation of shorter simulations with synchronized node start offsets yield optimization potential in terms of observing large frame latencies?

RQ2. Does randomization of node start offsets in aggregated short simulations enable optimizations beyond the synchronized case?

RQ3. Is the short simulation duration an important factor in the efficiency of the aggregation approach?

These research questions aim to evaluate whether aggregating short simulations can yield benefits over long simulations. Two aggregation alternatives were investigated per network configuration, and a total of five different network configurations were considered. One aggregation variant considers synchronized node start offsets, that is, all node start offsets are set to zero, and only the random seed is modified for each short simulation. The other variant uses different node start offsets uniformly sampled from a fixed range. The experiments serve to investigate whether either of these methods can yield improvements over the baseline solution of running a single long simulation. The system model will be explicitly described in detail in the following, and the experiment's setup and evaluation performed in the related publication will be presented subsequently.

3.2.1 System Model

This section will define the system model based on the networking concepts introduced in Chapter 2.2. The system model will defined to the degree that it can be reused in Chapters 4 and 5 without changes.

The core elements of the system model and simulation will be formalized to provide a complete understanding of the problem statement. However, the more intricate elements involved in the simulation, like, for instance, packet headers and the functioning of the

QoS mechanisms, will not be formalized as these details are not relevant for the understanding of the aggregation approach. These parts of the simulation will be treated as a black-box to keep the approach as generic as possible and avoid confusion.

A system model can be defined as a tuple

$$\mathcal{M} := (N, L, F, R, C)$$

where,

- N is the set of nodes, composed of the set of end-nodes E and the set of switches B
- L is the set of links that connect nodes from N
- F is the set of data flows f that define the traffic characteristics of communications generated by a sender and received by one (uni-cast) or multiple (multi-cast) end-nodes of the system
- R is the set of routes $r_e \in R$ which define, for each traffic flow f, the path that packets generated for flow f and reception by an end-node $e \in RECV(f)$ travel through the network
- $C_{\mathcal{M}}$ is the system clock defining a continuous global time

A traffic flow f represents a sequence of packets $\{ p_x(f) \mid x \in \mathbb{N} \}$ generated based on the flow properties, where x denotes a running sequence number that allows to uniquely identify each packet of a data flow. $SEND(f) \subset E$ denotes the set containing the sender end-node from which packets of flow f are sent. $RECV(f) \subset E$ denotes the set of end-nodes that receive packets sent by flow f.

Assumptions and Constraints

- The network topology and the routing of the flows remain static throughout the system's operation span. This is typical in the case of hard real-time systems as it is imperative to allow the analysis of the system behavior at design time.
- The packets generated by the traffic flows adhere to one of the flow types introduced in Chapter 2.2, periodic, sporadic, sporadic burst or dialog-based. All scheduling times are determined based on the local time of the sender node but expressed in terms of global reference time. Thus, the absolute generation times depend on the node clock drifts and other factors like node start offsets and frame offsets.
- The size of generated packets is fixed or upper-bounded.

- Local clocks of end-node are subject to a constant clock drift, determined in a range of up to 200ppm, determined as an acceptable bound in automotive applications in [80]. Clock drifts that vary over time as potentially encountered in real systems are not considered.
- The QoS mechanisms that determine how packets are handled in network bridges are a combination of FIFO, FP/FIFO or Credit-Based Shaping, as defined in Chapter 2.2.
- No component failures or transmission errors occur.

Concept of Time and Simulation States

The system time is given by a system clock $C_{\mathcal{M}}$, an absolute clock in the Newtonian sense. This means that we do not consider relativistic or spatial effects. The clock is defined as an ordered set of instants $C_{\mathcal{M}} := \{ t \mid t \in \mathbb{R} \}$ with the unit of one second and arbitrary precision. Every end-node possesses its own local time, given by a clock that can progress independently and at a pace relative to the system clock. This relative pace is described in terms of clock drifts as introduced in Chapter 2.2 and formalized later in this chapter. The start of the system denotes the origin of time $t_0 = 0$ and all times are expressed in terms of the global clock to avoid confusion.

The progression of real-time networking systems in terms of simulation can be described as a (possibly infinite) set of simulation states of model $\mathcal{M}$ as

$$S = \{ S(\mathcal{M}, t) \mid \forall t \in C_{\mathcal{M}} \}$$

where

- $S(\mathcal{M}, t)$ is the simulation state of model $\mathcal{M}$ at time instant t.

As discussed in chapter 2, discrete-event simulation only considers discrete changes that happen to the state of the network in the form of events. Consequently, for the set of simulation states S, multiple time instants $t \in C_{\mathcal{M}}$ may lead to equal simulation states.

Worst-Case End-to-End Latencies (WCTT)

End-to-end latencies of a flow packet p, send by an end-node $s \in SEND(f)$ received by an end-node $e \in RECV(f)$ are denoted by

$$TT(p, e) := t_{recv}(p, e) - t_{sched}(p, s)$$

where

- $p := p_x(f)$ is a packet with sequence number $x \in \mathbb{N}$ generated according to a flow $f \in F$

- $t_{recv}(p,e) \in C_{\mathcal{M}}$ is the instant of reception of packet p by receiver end-node $e \in RECV(f)$ according to the system clock $C_{\mathcal{M}}$
- $t_{sched}(p,s) \in C_{\mathcal{M}}$ is the instant when the packet is scheduled to be sent by the sender node $s \in SEND(f)$ according to the system clock $C_{\mathcal{M}}$

If there is a packet in transmission from the sender node, it may happen that the scheduling time t_{sched} and the actual send time t_{send} differ from each other. It generally holds true that $t_{sched} \leq t_{send} \leq t_{recv}$. While the computation of the end-to-end latencies only involves t_{sched} and t_{recv}, it implicitly integrates different delays as detailed in Chapter 2.2 caused by transmission mechanisms and other simultaneous traffic that delays the packet and can influence t_{send} and to a greater effect t_{recv}.

To compute the worst-case end-to-end latency $WCTT(f,e)$, also known as worst-case traversal time (WCTT), for a flow f and a destination end-node $e \in RECV(f)$, which is caused by one or multiple traffic scenarios that maximize the end-to-end delay $TT(p_x(f),e)$, we can derive the following formula:

$$WCTT(f,e) := max(\ TT(p_x(f),e) \mid \forall x \in \mathbb{N})$$

where $f \in F$ is a flow of the system and $e \in E$ is an end-node that receives packets $p_x(f)$ of flow f.

Thus, a solution $S_{wctt} \subset S(\mathcal{M},t)$ to the problem of maximizing observed end-to-end delays, as introduced earlier, can be formalized as finding the subset which contains the simulation states that maximize the observed end-to-end delays $TT(p_x(f),e)$ for all flows $f \in F$ and for all destination end-nodes $e \in RECV(f)$. Each such simulation state is observed at a time instant $t \in C_{\mathcal{M}}$ at which a packet $p_x(f)$ is received for which the observed end-to-end latency $TT(p_x(f),e)$ is maximal.

A solution can thus be formalized as:

$$\begin{aligned} S_{wctt} := \{\ S(\mathcal{M}, t_{max}(f,e)\ ) \mid \forall f \in F, \forall e \in RECV(f)\ \exists x \in \mathbb{N} : \\ t_{max}(f,e) = t_{recv}(p_x,e) \wedge TT(p_x(f),e) = WCTT(f,e)\ \} \end{aligned}$$

where $t_{max}(f,e) \in C_{\mathcal{M}}$ is the time instant at which a packet of flow f is received by end-node $e \in RECV(f)$ that yields a maximized delay.

3.2.2 Free and Fixed Parameters Impacting the WCTT

This section will analyze the effects of different parameters on the observed flow traversal times and which other factors contribute to it. In the general case, some of these parameters are considered free and can be controlled without breaking the simulated model's validity, while others are fixed and dictated by the system that is modeled. The parameters have been introduced conceptually in Chapter 2.2. Next, they will be formalized in the context of the system model, and their effects on the WCTT will be discussed.

The free parameters include, for instance, node start offsets and clock drifts. They can be adjusted within certain constraints to help explore the simulation space more effectively, which is the essential underlying idea of this thesis.

The fixed parameters include things like flow properties, link speed, routing, etc. These parameters define the properties of the system under simulation and must not be changed to maintain the validity of the simulation model.

Node Start Offsets

The node start offsets (NSO), as introduced in chapter 2.2, are free simulation parameters and define an offset for each end-node $e \in E$ with respect to the origin of time $t_0 \in C_{\mathcal{M}}$. These offsets define when an end-node is ready initially to start sending data. The node start offset for a node e is a time instant and is denoted by $NSO(e) \in C_{\mathcal{M}}$.

In real systems, these node start offsets are typically non-deterministic and can not be controlled. As a consequence, it makes sense to explore these node start offsets to achieve representative simulation results with respect to the system that is simulated.

Node start offsets implicitly impact how flow packets interact with each other during transmission from their sender to the receiver end-node and can thus be exploited in simulation to indirectly influence the observed end-to-end latencies of flows. Additionally, explicitly controlling them can be helpful in generating specific unfavorable scenarios.

Node Clock Drifts

Node clock drifts (CD) are free parameters used to model imperfections of clocks in real systems, as introduced in Chapter 2.2. We formally denote clock drifts as $CD(e)$ for each end-node $e \in E$. They represent a scalar value that describes the difference in velocity of the local clock C_e of an end-node e in comparison to the system clock $C_{\mathcal{M}}$. The local time instants can be converted to a global reference clock time as follows:

$$t_{C_e} = t_{C_{\mathcal{M}}} * (1 + CD(e))$$

where t_{C_e} is the time instant expressed according to the local node clock and $t_{C_{\mathcal{M}}}$ is the time instant according to the global system clock.

Clock drifts do not have a direct impact on the traversal times. However, they cause the temporal alignment of different nodes to drift apart from each other over time. This means even if a system of nodes that is supposed to send data periodically in the same interval, the relative sending times of packets originating from different nodes may drift apart over time due to the clock drifts, as observed in Figure 2.3.

Clock drifts can impact the hyper-period of the system and can significantly influence the order in which simulation states are explored. They can further break the ergodicity property if chosen in a specific manner, harming the ability to explore all relevant simulation states from a single simulation. Proving or disproving ergodicity of simulation is beyond the scope of this thesis and will be assumed to be true as empirically observed in previous work by Navet, Seyler and Migge in [80].

Frame Offsets

Frame offsets (FO), as introduced in Chapter 2.2, allow to shift the instant t_{sched} at which the first frame of a flow $f \in F$ is scheduled with respect to the starting time $NSO(s)$ of the sender node $s \in SEND(f)$. The frame offsets of a flow $f \in F$ are denoted by $FO(f) \in \mathbb{R}$. The scheduling time of the first packet of a flow $f \in F$ can thus be expressed as

$$t_{sched}(p_0(f), s) = t_0 + NSO(s) + FO(f)$$

While, in general, frame offsets change the traffic characteristics of the system, in simulation, they can also be used to control the scheduling order of flows that are scheduled at the time instant. When steering the frame scheduling order is the goal, selecting very small frame offsets is essential to avoid impacting the traffic characteristics in a significant manner. In this dissertation, the frame offsets will be used for that purpose in Chapter 5 where the potential of intentionally exploring and optimizing the flow scheduling order will be investigated.

Simulation Random Seed

The random seed of simulators is a technical means to make stochastic effects during the simulation process deterministic and allows the generation of reproducible results. The effects of the random seed of the simulation software may indeed vary depending on the software implementation and how the seed is used internally. It can thus not be reasonably formalized without also formalizing the complete inner functioning of the simulation software.

One typical effect of the random seed is determining the sending order of packets scheduled on a node or the queueing order of packets received by a switch at the exact same time. Another application is the generation of varying delay components contributing to the switching delay inside the network bridges. Further, it is used for certain traffic types like sporadic flows, where the delay between sending two consecutive packets is randomly generated from a certain distribution.

In the context of this work, the random seed is used to explore different simulation trace variations based on these effects. For instance, the random seed must be changed when running multiple long simulations with equal starting conditions, like node start offsets and clock drifts. Otherwise, the results would be equal for all simulation instances.

3.3 Experiments

The setup behind the experiments is based on a simulation service that provides the ability to load a certain network configuration and run simulations in parallel based on differing sets of input variables. The network configuration is only loaded once after the simulation engine is initialized, as the fixed configuration parameters remain constant, and only free parameters are adjusted for each simulation instance.

The input variables applied in this chapter comprise the simulator random seed, and the node start offsets (NSO) and clock drifts (CD) for each node. The outputs of the simulation service are the maximal traversal times observed during the simulation for every flow and receiver end-node. The experiment script generates a set of randomized node start offsets from the node start offset range, which was chosen to explore the close NSO region around the synchronized case. The node clock drifts are drawn randomly from a [0, 200]ppm range but remain constant across all simulation instances in this chapter.

3.3.1 Simulation Host

The host machine on which all experiments of this dissertation were performed features an AMD Ryzen 5950X 16-core CPU @4600Mhz, 128GB DDR4 RAM and an NVMe hard drive to store the data. The operating system used for the experiment host machine is Ubuntu 22.04 LTS. The Java virtual machine used is provided by OpenJDK 18.0.2.

3.3.2 Test Cases

Five network configurations based on three different topologies from industrially relevant applications were used to evaluate the aggregation approach. The different topologies and their corresponding configurations have the following properties:

- **Space Launcher:**
  The design of the space launcher topology is shown in Figure 3.3. It uses a highly connected grid-like structure and is constructed of a total of 18 nodes, 18 switches, which are integrated with an end-node each and are connected by a total of 24 links or 42 links when considering the hop between the switches and the integrated end-node. For the space launcher, a single configuration that uses 4 fixed priority levels (FP/FIFO) for traffic management is considered. The configuration includes a total of 100 multi-cast frame flows, that result in a total of 985 flow receptions.

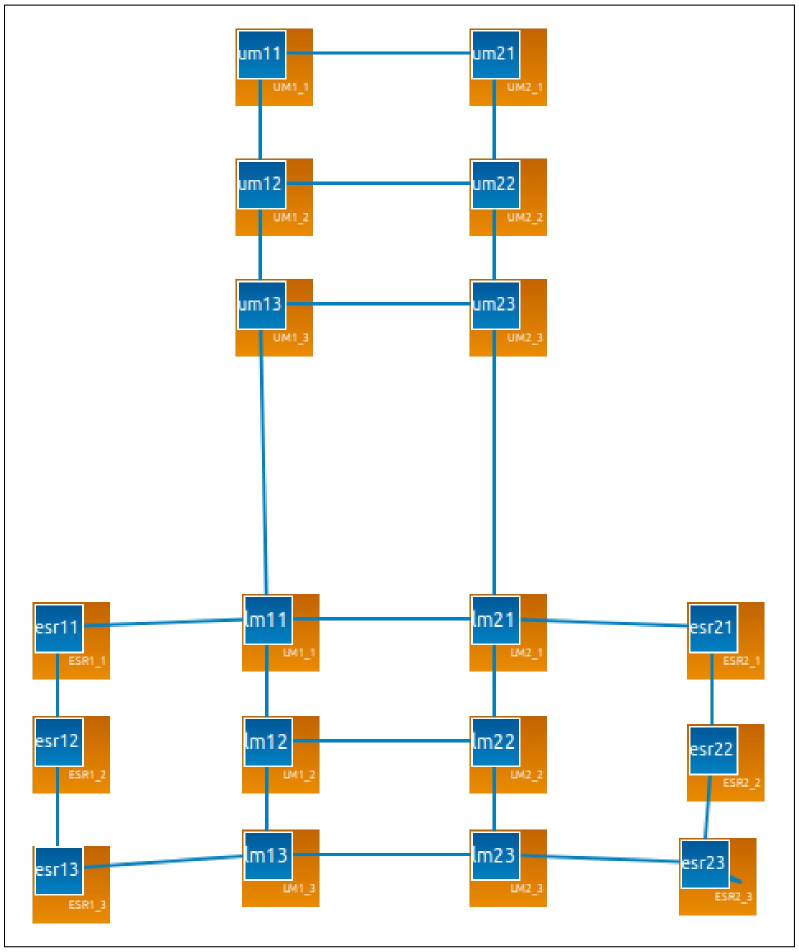


Figure 3.3: **Space launcher system topology.** Illustration as presented in publication [56].

- **Automotive:**
  The topology illustrated in Figure 3.4 represents an embedded network from an automotive application. It is comprised of 14 end-nodes that are connected to 5 switches via a total of 18 links. The configurations on this topology each include 58 flows that result in a total of 70 receptions. Two different configurations on this topology are considered, one configuration applying 4 fixed priorities (FP/FIFO) and the other configuration including Credit-Based Shaping (FP/FIFO + CBS) in addition to the 4 priorities. The traffic flows are categorized into 19 periodic ”Command&Control” flows, 10 periodic ”Audio” flows, 11 periodic burst ”Video” flows, 6 periodic ”Best Effort” flows, and 4 TFTP flows where each flow is subdivided into a request flow (RRQ), a response data flow (DAT) and an acknowledgment flow (ACK). For the TFTP traffic, the DAT flow is triggered as a response to the request flow, and the ACK flow is triggered as a response to the reception of a DAT packet, as described in Chapter 2.2.

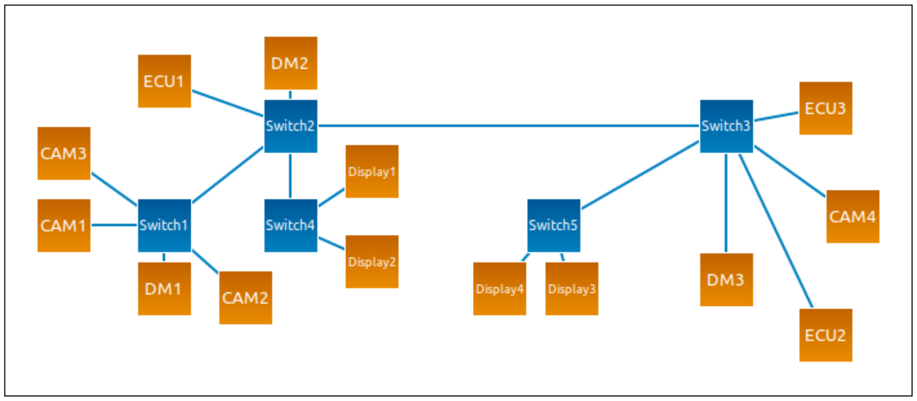


Figure 3.4: **Automotive system topology.** Illustration as presented in publication [56].

- **Avionics:**
  The configurations for the avionics use case are based on the medium-sized ring topology shown in Figure 3.5. This topology is based on 52 nodes that are connected via 4 switches with a total of 57 links. Each configuration on this topology features 453 multi-cast flows, resulting in a total of 3214 flow receptions. For this topology, one configuration that uses first-in-first-out traffic management (FIFO) has been defined, and one configuration that uses 5 fixed priorities (FP/FIFO). The traffic consists exclusively of sporadic flows.

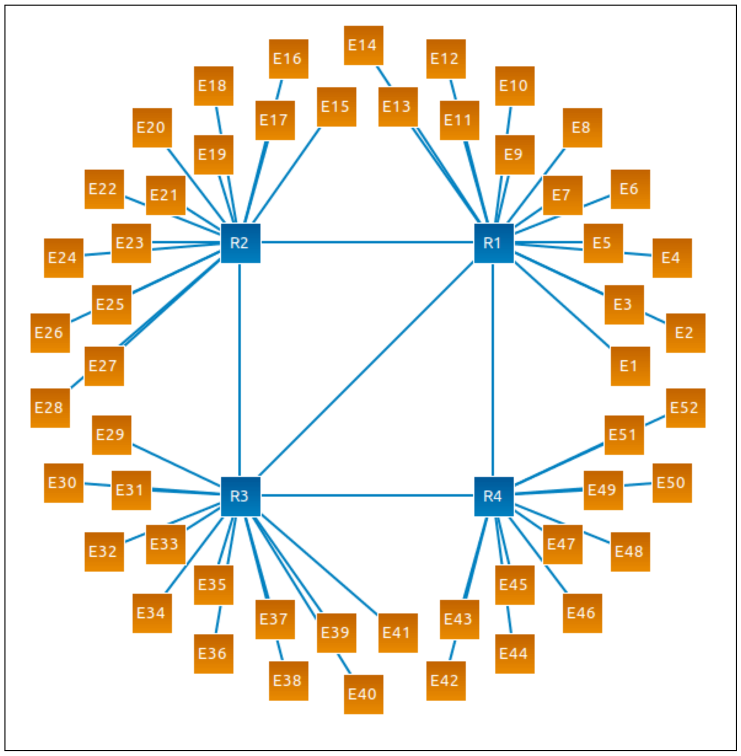


Figure 3.5: **Avionics system topology.** Illustration as presented in publication [56].

The data of the different configurations described above is summarized in Table 3.1 for simple side-by-side comparison. Further details on the distributions of flow receptions, frame sizes, periods/BAG and deadlines are given in Appendix B.

3.3.3 Experiment Design

To address the three research questions defined in section 3.2, a total number of twelve experiments were conducted across the previously presented network configurations. Table 3.2 gives an overview of the experiments that were conducted. The performance of aggregated short simulations is compared to long simulations for each network configuration. For each comparison, a total of 100 hours of simulation was generated. It means that a total of 100 hours of short simulations are aggregated and compared to a single long simulation of 100 hours of simulation time. To achieve a statistically representative result, many such simulation pairs are generated.

The experiment flow is divided into the following steps:

1. Generate a set of many short simulations.
2. Applying bootstrapping (see Appendix A.1) to resample from the set of short simulations and create aggregations each of 100 hours of total simulation time.
3. Run several 100-hour-long simulations.
4. Generate all combinations of aggregations and long simulations.
5. For each pair, compute the per-flow difference between long and aggregated simulation results.
6. Compute the per-flow average over all short-long pair differences.
7. Represent the distribution of relative performance between long and aggregated short simulations by plotting the data as a box plot for evaluation.

Following this workflow, for each experiment instance, one hundred aggregations of 100 hours of simulation time were resampled and pairwise combined with a total of 16 long simulations of 100 hours of simulation time each. So, each reported experiment is based on 1600 sets of comparisons.

For each network configuration, two sets of experiments were performed. One set of short simulations is run using synchronized node start offsets. The other set of short simulations is run by sampling the node start offsets from a continuous uniform distribution over a fixed range with a selected upper bound of 0.1ms to achieve NSO close to the synchronized case. For each short simulation, the random seed of the simulator is changed to enable the exploration of different flow scheduling orders for flows originating from the same node, as described before. Randomized clock drifts were applied, which remained constant across all experiment simulations to exclude clock drifts as an impacting factor.

Table 3.1: **Characteristics of the network configurations for RQ1-RQ3.**
For each of the avionics and automotive topologies, two configurations are defined that each use different Quality-of-Service (QoS) traffic management mechanisms. FIFO means first-in-first-out traffic management. FP/FIFO(4) means traffic with 4 fixed priority levels and FIFO handling of flows of equal priorities. The TFTP flows each consist of three component flows (request ”RRQ”, data ”DAT” and acknowledgment ”ACK”). Table as presented in publication [56], with adjustments.

Topology	# nodes	# switches	# links	# flows	# receptions	QoS	flow types
Space Launcher	18	18	24	100	985	FP/FIFO(4)	21 C&C (periodic) 78 Telemetry (periodic) 1 Video (periodic burst)
Avionics	52	4	57	453	3214	FP/FIFO(5) FIFO	453 Uncategorized (sporadic)
Automotive	14	5	18	58	70	FP/FIFO(4) + CBS FP/FIFO(4)	19 C&C (periodic) 10 Audio (periodic) 11 Video (periodic burst) 6 Best Effort (periodic) 4 TFTP (each ACK+DAT+RRQ)

Table 3.2: **Experiment parameter settings.**
QoS stands for Quality of Service. ST stands for simulation time t_{sim}. NSO stands for Node Start Offsets. FIFO stands for first-in-first-out. FP/FIFO stands for fixed priorities using FIFO management among packets of the same priority. A NSO range of 0.0 denotes offsets synchronized to zero. Table 3.1 shows the number of priorities used. CBS stands for Credit-Based Shaping. The table is based on the publication [56].

ID	Topology	QoS	ST[s]	NSO range [ms]
EX1	Space Launcher	FP/FIFO	30	0.0
EX2			30	$[0.0, 0.1]$
EX3	Avionics	FIFO	30	0.0
EX4			30	$[0.0, 0.1]$
EX5		FP/FIFO	30	0.0
EX6			30	$[0.0, 0.1]$
EX7	Automotive	FP/FIFO + CBS	30	0.0
EX8			30	$[0.0, 0.1]$
EX9		FP/FIFO	30	0.0
EX10			30	$[0.0, 0.1]$
EX11			120	0.0
EX12			120	$[0.0, 0.1]$

The simulation time of 30 seconds and the node start offset range of $[0.0, 0.1]$ milliseconds were empirically determined to return reasonable results from the automotive configuration with FP/FIFO and applied without further modification to the other test cases. The experiments in this chapter were designed to observe how well simulation aggregation would perform without optimizing hyper-parameters, like simulation time and node start offset range. After observing that 30 seconds of simulation time performed the worst on the automotive FP/FIFO case, the decision was taken to repeat this experiment with 120 seconds of simulation time for comparison.

For each experiment, a set of short simulations is generated and resampled using bootstrapping simulation to one hundred different aggregations of 100 hours of short simulations, and sixteen long simulations with a 100-hour simulation time are run. Each aggregation is combined pairwise with each long simulation to yield a total of 1600 simulation pairs. Additional information on bootstrapping and how it was used in the experiments can be found in Appendix A.1 and in [56].

3.3.4 Results

The results of the experiments are shown in Figures 3.6, 3.7, 3.8, 3.9 and 3.10 as boxplots, and are discussed categorized per research question in the following. The results are presented in a slightly different structure here but are recited from the findings in publication [56], where also the figures originate from. Appendix A.4 explains how boxplots display information and how they can be interpreted.

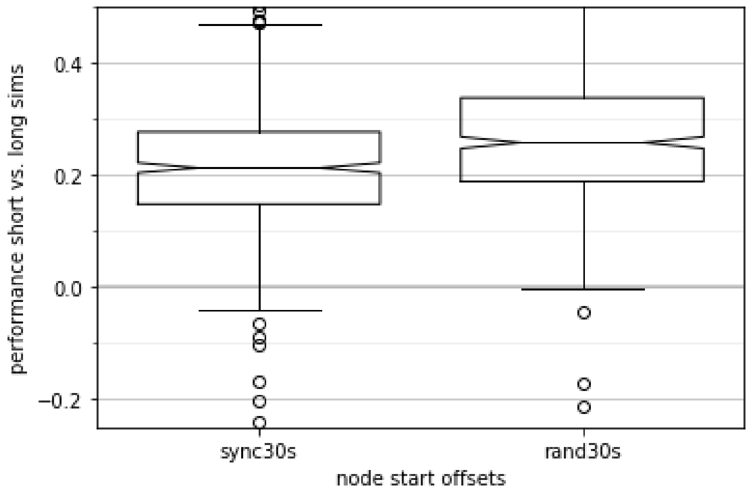


Figure 3.6: **Results for RQ1 and RQ2 - Space Launcher topology with FP/FIFO.** Results for experiments *EX1* and *EX2*. The y-axis shows the relative difference (as a decimal fraction) between the maximal observed end-to-end delays with short and long simulations over all flows (averaged over all experiment instances). Short simulations here clearly outperform long simulations in both, synchronized (median +21.3%) and non-synchronized (median +25.8%) cases. Figure as presented in publication [56].

Results for RQ1: Performance of synchronized NSO aggregation

To answer research question RQ1, experiments EX1, EX3, EX5, EX7, EX9 and EX11 were conducted. The research question addresses whether aggregating short simulations using synchronized NSO, like done by the baseline approach can yield any improvements over long simulations. Each of these experiments aggregates short simulations of 30 seconds simulation time (120 seconds for EX11) based on synchronized node start offsets and compares the per-flow distribution with the results of long simulations according to the baseline solution.

The results of these experiments are shown in the boxplots denoted by ”syncXs” on the x-axis of the result Figures 3.6, 3.7, 3.8, 3.9, and 3.10 for each of the configurations, where X is either 30 or 120 for the corresponding simulation time in seconds. The y-axis of each plot describes the distribution of the per-flow differences between the aggregated short simulations and the long simulations.
To summarize, for RQ1, we observe that for all, except the 30-second experiment on the automotive configuration with FP/FIFO, the aggregated short approach, on average, im-

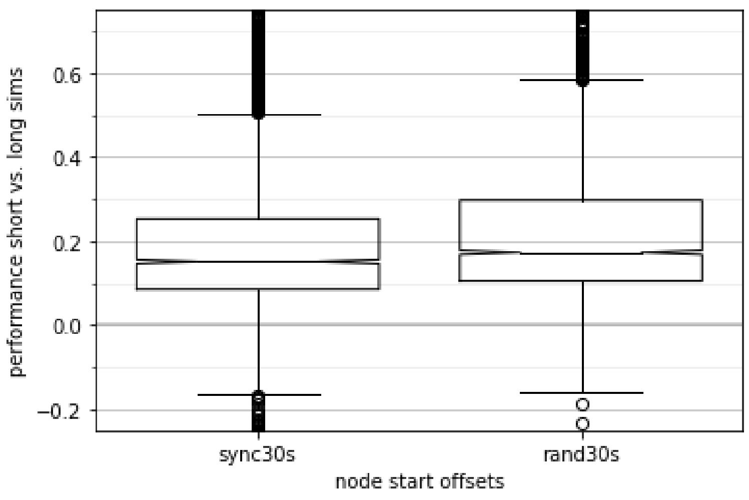


Figure 3.7: **Results for RQ1 and RQ2 - Avionics topology with FIFO.**
Results for experiments *EX3* and *EX4*. As seen by the position of the boxes (*i.e.*, interquartile range), short simulations perform better than long simulations on average. The median of the synchronized case shows an improvement of about +15.2% and the uniform sampling of NSO shows an improvement of +17.4%. The results figure was presented in [56].

proved over the baseline approach using long simulations. The concrete mean differences for aggregated short simulations compared to long simulations are as follows:

- +21.3% median improvement for the space configuration (EX1).
- +15.2% improvement for the avionics configuration with FIFO (EX3).
- +10.4% improvement for the avionics configuration with FP/FIFO (EX5).
- +4.0% median improvement for the automotive configuration with FP/FIFO + CBS (EX7).
- -1.6% median decline for the automotive configuration with FP/FIFO (EX9) with 30 seconds simulation time.
- a slight improvement of +0.04% for the automotive configuration with FP/FIFO (EX11) with 120 seconds of simulation time.

Overall, the results are overwhelmingly positive for simply breaking down the long simulations into shorter ones. This indicates the potential of exploring flow scheduling orders as the random seed was the main component that differed between the short and long simulations in these experiments.

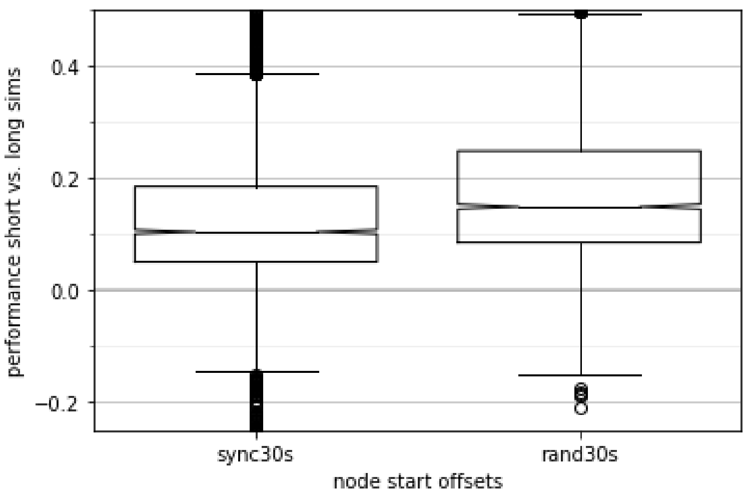


Figure 3.8: **Results for RQ1 and RQ2 - Avionics topology with FP/FIFO.** Results for experiments *EX5* and *EX6*. Short simulations again outperform long simulations as the medians are +10.4% higher for the synchronized case and +14.9% higher for the uniformly sampled NSO. Results figure as presented in [56].

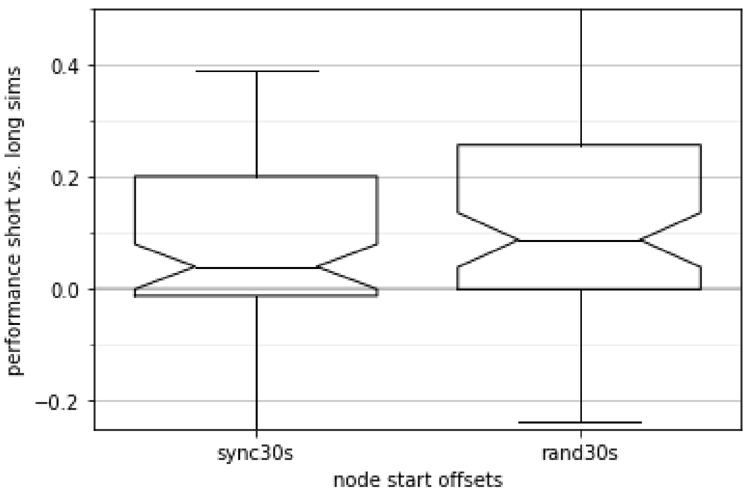


Figure 3.9: **Results for RQ1 and RQ2 - Automotive topology with FP/FIFO+CBS.** Results for experiments *EX7* and *EX8*. Aggregation of short simulations performs better for about 57.1% (synchronized case) and 72.9% (non-synchronized case) of the flows. Figure as presented in [56].

Results for RQ2: Further improvements via randomized node start offsets

This research question aims to evaluate whether randomizing node start offsets of short simulations in the range of $[0.0, 0.1]$ms can further improve the observed performance over using synchronized NSO. To evaluate the performance difference of using the ran-

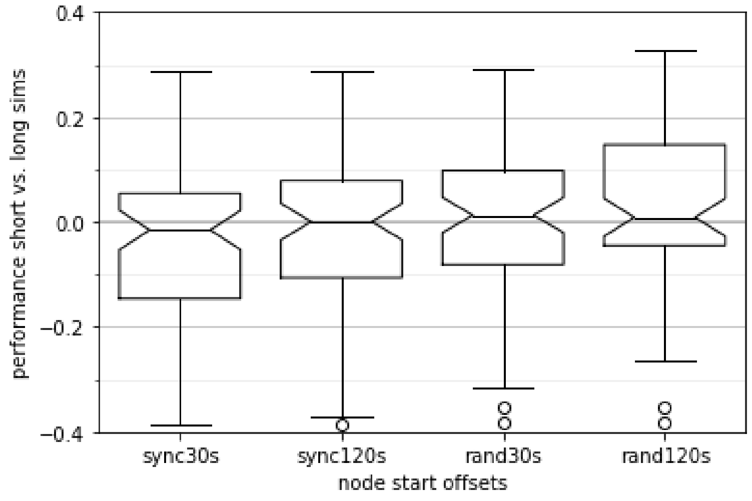


Figure 3.10: **Results for RQ1 and RQ2 - Automotive topology FP/FIFO.** Results for experiments *EX9*, *EX11*, *EX10* and *EX12*. The difference between short and long simulations is small here, as all medians are around zero. A small increase in the relative performance of short simulations can be observed across the four experiments. Figure and description as presented in [56].

domized node start offsets, the result Figures 3.6, 3.7, 3.8, 3.9 and 3.10 are consulted again. This time, the box plots denoted by ”randXs” on the x-axis are considered. They represent the experiment results of the simulation aggregation with randomized node start offsets.

To summarize, for RQ2, we observe that, for all test cases, the mean performance of the aggregated simulations using randomized node start offsets increased. This is true in comparison to the long simulations but also in comparison to the aggregated simulations using synchronized NSO. The concrete mean differences for aggregated short simulations with randomized NSO are as follows:

- +25.8% median improvement over long simulations for the space configuration (EX2). This is an improvement of 4.5% over the aggregation with synchronized NSO (EX1).
- +17.4% improvement for the avionics configuration with FIFO (EX4). This is an improvement of +2.2% over the aggregation with synchronized NSO (EX3).
- +14.9% improvement for the avionics configuration with FP/FIFO (EX6). This represents an improvement of +4.5% over the aggregation with synchronized NSO (EX5).
- +8.8% median improvement for the automotive configuration with FP/FIFO + CBS (EX8), which represents again a +4.8% improvement over the aggregation with synchronized NSO (EX7).
- +1.3% median improvement for the automotive configuration with FP/FIFO (EX10)

with 30 seconds of simulation time. This is an improvement of +2.9% over the aggregation with synchronized NSO (EX9).

- A slight improvement of +0.86% for the automotive configuration with FP/FIFO (EX12) with 120 seconds of simulation time. This represents a relative improvement of +0.82% over the aggregation with synchronized NSO (EX11).

Overall, these results represent a significant improvement for applying a method as simple as uniformly sampling node start offsets from a fixed range, which, in addition, was not optimized in any way.

Results for RQ3: Significance of simulation time and NSO range

This research question aims primarily to observe whether simulation time plays an important role in the specific performance of the aggregation approach. The results related to this research question are found in Figure 3.10. This figure enables the comparison of experiments with similar starting conditions but differing simulation times, concretely 30 seconds and 120 seconds per short simulation. It needs to be recalled that both types of experiments were accumulated for a total of 100 hours of simulation time. Thus, the number of simulations at 120 seconds per experiment is accordingly smaller, reducing the exploratory power of the aggregation method.

For the synchronized node start offsets, the median performance compared to long simulations increases from -1.6% to +0.04% for 30-second simulations, thus a relative improvement of 1.64%. For the randomized node start offsets, however, the median performance difference slightly decreases from +1.3% down to +0.86%, thus a relative decrease of -0.44%.

An interesting effect to observe is the shift of the overall bulk of the distributions across the different experiments. For the longer 120-second simulations, the overall performance across the distribution of flows goes up for both the synchronized and randomized NSO experiments compared to their 30-second counterparts.

3.4 Conclusion

Considering the results of the experiments for RQ1, it can be concluded that simply splitting the long simulations into shorter ones while retaining synchronized node start offsets already yields a significant improvement over running long simulations. This suggests the importance of exploring different flow scheduling orders, which is the primary effect of changing random seeds for the aggregated short simulations while clock drifts and node start offsets remained constant for these experiments. Additionally, the aggregation approach allows for a substantially increased parallelization, which is not the primary focus of this dissertation but a beneficial side-effect that can have highly important implications and will be discussed subsequently.

The research question RQ2 of whether randomizing the node start offsets is beneficial can also be answered positively. In fact, the sampling range was not optimized in any manner, but the observed improvements were significant and provided insight into the untapped potential of the aggregation approach. This potential will be further explored in the two following chapters. In Chapter 4, the effects and settings of simulation time and node start offset selection will be investigated more systematically, and in Chapter 5, simulation time will be minimized to increase the exploratory power of the aggregation approach, and node start offsets and flow scheduling order will be optimized.

Regarding RQ3, the results are inconclusive as the median performance did not show a consistent improvement. However, it could be observed that the longer simulations, together with randomized node start offsets, were beneficial for the overall performance distribution of the flows, even though the median performance slightly decreased. At the time of publishing, some effects of parameters were poorly understood and were left for further investigation in future research. Consequently, based on more recent insights, it can be said that the simulation was mostly limited by the chosen node start offset range in this chapter.

The fact that the same randomly determined clock drifts were applied for all short simulations in this experiment further decreased the exploratory potential of the aggregated short simulations as the applied simulation times were still significantly larger than the simulation times explored in Chapter 5. It appeared reasonable to limit the effects of the clock drifts for the experiments conducted in this chapter so as to reduce the number of free variables. As a consequence, the exploratory effect of clock drifts on longer simulations was not carried over to the short simulations in this case. Thus, by randomizing the clock drifts for each short simulation, the results of the aggregations could likely have been improved further.

3.4.1 Intuition on the Effectiveness of Simulations Aggregation

The intuition behind the effectiveness of simulation aggregation with differing starting conditions is based on the observation that during simulation, instants with high traffic are generally sparse causing a reduced number of interferences. This means that many computational resources are spent in areas of the simulation space that often will not allow moving closer to the actual worst-case traversal times of the flows. While it is assumed that randomized node clock drifts allow us to explore the whole simulation state space via ergodicity, they only allow doing so slowly with progressing simulation time. To increase the chances of observing higher end-to-end delays for a flow, it is imperative to cause scenarios where as many flows as possible interfere with the transmission inside the switches on the flows route.

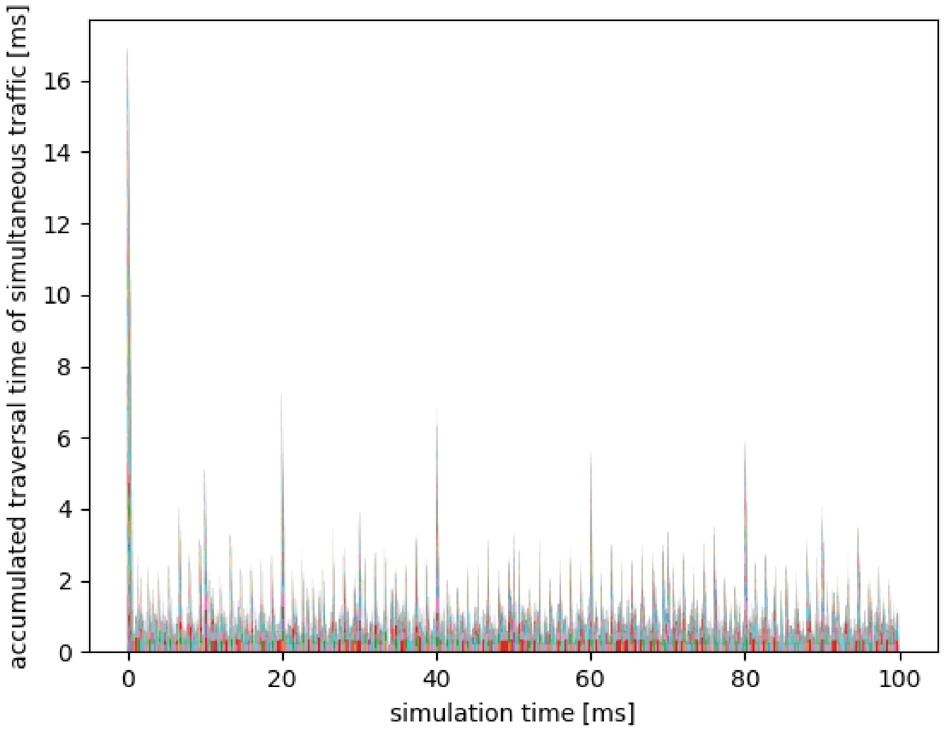


Figure 3.11: **Simultaneous traffic for 100ms simulation of the synchronized NSO baseline solution on the automotive topology with FIFO traffic.**
The x-axis shows the simulation time. The y-axis represents the stacked height of traffic simultaneous in transit in the network. The contribution of each flow is determined by the end-to-end delay of the packet in transit. As can be seen, the accumulation of simultaneous traffic peaks at the start and shows some lower peaks later, with a lot of sparse low-traffic areas in between.

To illustrate the effects of the synchronized NSO case, Figure 3.11 shows the stacked simultaneous traffic of a simulation trace of 100 milliseconds simulation time. As can be observed, the most simultaneous traffic, hence roughly speaking, the highest probability of observing overall high traversal times, lies at the start of the trace where all flows start scheduling packets simultaneously. However, high simultaneous traffic does not necessarily translate exactly to the highest traversal times for all flows. For that purpose, in Figure 3.12, observed traversal times for a selection of flows from the same trace are shown. As can be observed in that figure, many flows experience the highest delays during the initial instants of high simultaneous traffic. However, it can also be seen that some flows experience higher delays at later instants. This motivates the exploration of node start offsets distant from the synchronous case.

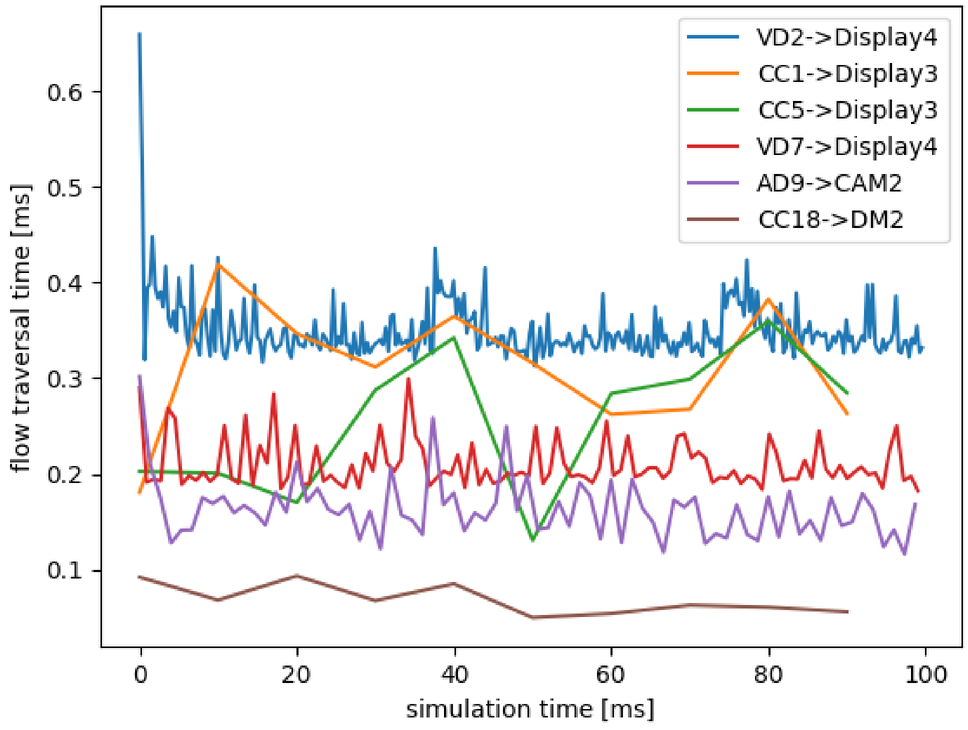


Figure 3.12: **Per-flow traversal times for 100ms simulation of the synchronized NSO baseline solution on the automotive topology with FIFO traffic.**
The x-axis shows the simulation time. The y-axis represents the traversal time of a packet flow. Since this test case includes 58 flows, a selection of 6 randomly selected flows is shown; otherwise, the lines would be impossible to discern. Overall, it can be observed that most flows generate their highest traversal times during the first few milliseconds, and after the observed traversal times fall off for the most part.

While the traditional approach, as discussed in the previous section, manages to cause a good amount of flow interferences in their first switches, it also reduces the chance to observe higher interferences at later switches on the routing path. To increase the probability of this, we can slightly shift the node start offsets of the nodes, which causes other flow packets to arrive sooner or later at the switches through which our flow of interest travels.

An alternative way of understanding the potential benefits of simulation aggregation via node start offsets and flow scheduling order is illustrated in a dummy example of a simulation state system in Figure 3.1. This example aims to illustrate how the starting state and trajectory of simulation traces can change with different starting conditions, yielding scenarios where higher end-to-end delays are observed earlier or later. When starting conditions are chosen carefully, the number of visited simulation states required to observe higher end-to-end delays can be reduced, particularly in simulation aggregation.

3.4.2 Unreported Exploratory Experiments

As discussed in this chapter, the effects and relation between simulation time and node start offset range were investigated empirically by various preliminary exploratory experiments. The results of these exploratory experiments are not reported here as they were not deemed conclusive enough, but they helped inform a deeper understanding of the parameters and properties of flows that are reported in this work. For instance, some experiments were conducted to explore the effects of randomizing clock drifts instead of node start offsets. In some cases, these experiments yielded similar results to randomizing the node start offsets but at diminishing effectiveness the shorter the simulation time. This stems from the fact that clock drifts do not allow to cause an immediate impact on the interferences generated in the switches but need time to develop their effect. By using node start offsets, on the other hand, we can more precisely steer how flows interact in switches at the beginning of a simulation as observed in Section 3.4.1.

Another type of exploratory experiment was conducted to determine the effects of the simulation time. Specifically for the prioritized cases higher traversal times could be observed for certain flows by shorter simulations, while other flows would display higher traversal times with longer simulations. No exact patterns could be identified to consistently observe this effect. It could be an indication that the chosen node start offset range was not appropriate and needed to be supported by the exploratory effect of clock drifts over time. In the following chapter, strategies to determine reasonable simulation times and node start offsets are derived and evaluated.

3.4.3 Parallelization Potential

In this chapter, it was concluded that the aggregation approach yields a significantly increased potential for parallelization as a beneficial side effect. Even though the increased potential is not formally researched in this dissertation, it is important to discuss and investigate this significant advantage.

To illustrate the achievable parallelization speedup, a thought experiment is conducted. Let's assume we can run a long simulation of 100 hours of simulation time for a duration of 10 hours, and a 1-hour simulation of the same configuration for a duration of 6 minutes. This would represent the same simulation speedup factor of 10 and may be realistic according to Figure 4.1. Now if assuming perfect parallelism, the aggregation of 100h total simulation time composed of 1-hour simulations can effectively be run in a duration of about 6 minutes instead of 10 hours by running all 100 simulations in parallel. This would represent a parallelization speedup factor of 100. It needs to be considered, however, that the speedup factor drops when the simulation time is reduced too much, as will be explored in the next Chapter.

Further, since there is always some non-parallelizable overhead, the actual possible parallelization speedup factor would be reduced by this portion as defined in Section 2.4.4. The portion of the aggregation approach that can not be parallelized is difficult to determine as it involves multiple factors:

- The overhead of generating diverse starting conditions.
- The overhead of aggregating the simulation results.
- Potential difference in simulation duration due to diverse traffic scenarios, causing synchronization losses.
- The reduction of the simulation speedup factor for shorter simulation times, which essentially can be seen as a non-parallelizable overhead cost.

Conducting an empirical study that investigates the different components may be helpful to shed light on the true parallelization potential of the aggregation approach as presented in this dissertation and may help derive strategies to reduce each of the contributing overhead factors. This work is retained as a potential future work.

It needs to be emphasized though, that this parallelization speedup is separate from the aggregation speedup concluded in Chapter 4 and Chapter 5, as these speedup factors will represent the reduced number of (sequential) simulations required to produce similar WCTT bounds than long simulation. That means that those aggregation speedups and the speedup gained from parallelization can essentially be multiplied.

As the reduction of the simulation speedup factor is a result of simulation initialization overhead, it would be beneficial to develop network simulation software that is optimized for running very short simulations. As a result, this would significantly increase the efficiency of the aggregation approach even further, as running extremely short simulations allows for a greatly improved exploration potential as demonstrated in Chapter 5.

Chapter 4
Improved Parameter Choice for Simulation Aggregation

Setting the short simulation time and the node start offset range was left open in the previous chapter. It was concluded that the choice of simulation time and node start offsets play an important role in unlocking the potential of simulation aggregation. Thus, this chapter proposes and evaluates methods for setting the simulation time of short simulations in aggregation such that the simulation speedup factor is roughly maintained. This allows for a trade-off of exploring the simulation state space by considering different starting conditions and exploration resulting from clock drifts in longer simulations.

A meaningful choice of the starting conditions for the aggregated short simulations appears even more important, as explored and discussed in the previous chapter. To that end, an overlapping stratified sampling approach for node start offsets is proposed to improve the exploration of the starting condition space. The choice of overlapping stratified sampling will be presented, motivated, and evaluated based on test cases, which are similar or identical to those investigated previously. The methods proposed in this chapter are designed to be beneficial and easily applicable for practitioners who seek to apply the aggregation approach to their use cases.

Some of the material presented in this chapter is part of a paper ready for publication [54] and will be referenced accordingly. Several illustrations of results not present in the paper have been added.

4.1 Motivation and Context

As was concluded in the previous chapter, the simulation time and the node start offset range are imperative parameters to unlock the effectiveness and efficiency of the aggregation approach. While it was observed that longer simulation time naturally allows for observing higher traversal times due to the exploration based on the randomized clock drifts, empirical evidence has shown that leveraging the exploratory power of shorter simulation times can be preferential.

Generally speaking, the aim is to reduce the simulation time as much as possible in order to increase the exploration of starting conditions and thus cover a larger portion of the simulation state space, increasing the likelihood of observing higher end-to-end delays. There are, however, limitations to reducing the simulation time. These limitations have both theoretical and technical roots.

The theoretical limitation of the simulation time is defined by the minimal simulation time required to obtain an end-to-end delay value for every flow reception. The technical limitation is more implicit and is caused by a loss of efficiency due to decreasing simulation speedup with decreasing simulation time. When reducing the simulation time, the initialization cost of the simulation gains increasing significance in comparison to the resources spent on generating simulation data. The details of these limitations will be discussed in more detail in a dedicated section.

The contributions of this chapter are twofold. On one hand, an approach is described that allows to determine a minimized short simulation time, which roughly maintains the computational speedup factor of longer simulations while allowing for increased exploration at the same time. This method can be adjusted according to the constraints and requirements of the application. On the other hand, a stratified sampling approach is developed, which improves overall performance by better balancing the exploration of larger node start offset ranges and exploitation of the knowledge that narrow ranges around the synchronized case yield decently high traversal times in many cases, as observed in Chapter 3. Applying this sampling method allows to explore larger node start offsets that may be necessary to observe the highest end-to-end delays, without missing out on the benefits of the near-synchronous offset region.

4.1.1 Related Prior Work

The work presented in this chapter represents a direct continuation of the foundations provided by the previous Chapter 3. This section only focuses on the closest relevant prior work, as the broader context was already presented extensively in Chapter 2.

The improved approach for sampling the node start offsets (NSO) will be introduced later in this chapter. It relies on overlapping stratified sampling and was inspired by the insights on the general structure of unfavorable scenarios as observed on the trajectory approach proposed by Bauer, Scharbarg and Fraboul in [18] and [19], which was discussed in detail in Chapter 2. The approach they proposed in these works aims to generate transmission scenarios leading to ”critical instants” that yield high traversal times. However, their approach is limited to only sporadic flows, and was developed for FIFO and fixed priority (FP/FIFO) traffic only. To generate these unfavorable scenarios, flow scheduling instants are shifted and sorted according to heuristics informed by certain properties of the traffic management mechanisms. To be able to move the flows around, the work relies on the bandwidth allocation gap (BAG), which defines a minimal time between two consecutive emissions of packets of a flow, a core property of sporadic flows. As sporadic flows do not have a fixed periodicity, these exact moments when a flow is scheduled are adjusted by the analysis, to produce unfavorable scenarios leading to high interferences along the flow path.

The underlying philosophy of the approach presented in this dissertation is similar. The node start offsets are used to control the movement and order of flow scheduling instants, aiming to increase the probability of observing scenarios that yield high end-to-end latencies. As node start offsets do not allow to impact the ordering of flows originating from the same node, as discussed before, the scheduling order will additionally be controlled via the means of minimal frame offsets in Chapter 5.

Moreover, within the existing body of literature, such as the work by Sifakis and David in [92], there is a prevalent consensus that the growing complexity of industrial systems, especially due to the continuous movement toward automation, necessitates the use of simulation as the primary means of validation for industrially relevant scenarios. In these applications, precise analytical approaches are often not available, or they are too pessimistic and thus not cost-efficient. This motivates the further exploration of enhanced validation methods based on simulation, such as the approach proposed in this dissertation.

4.1.2 Limitations in the Reduction of the Simulation Time

The simulation time of short simulations can reasonably only be reduced to a certain point without a significant loss of performance. However, two different limits need to be considered in practice: hard and soft limits.

The hard limit can be defined as the point below which the simulation time is too short to produce traversal times for every flow. While this may be desirable in some exceptional cases, for instance, when only focusing on a specific subset of flows, the usual case is to collect data on all flows. It is rather simple to define this limit theoretically, but it is practically difficult or often impossible to determine it precisely as it involves the true worst-case traversal times, which are unknown.

The lower limit for the simulation time can be defined as the maximal time it takes for each flow to transmit at least one packet (or burst of packets) after the system start by

$$t_{min} := max(NSO(SEND(f)) + FO(f) + WCTT(f, e) \mid \forall f \in F, \forall e \in RECV(f))$$

where:

$f \in F$ is a flow from the set F of all flows of a configuration
$NSO(SEND(f))$ is the node start offset of the sender node of flow f
$FO(f)$ is the node start offset of the sender node of flow f
$WCTT(f, e)$ is the actual worst-case traversal time of flow f reception in node e

This formula describes the time instant when the latest first packet (or burst) of a flow arrives at its destination, based on the flow sender's corresponding node start offset, frame offset, and worst-case delay. In this chapter, the frame offsets $FO(f)$ are assumed to be zero. As the node start offsets are generally not bounded, this formula is not very practical unless we define an upper limit on the possible node start offsets.

Additionally, since $WCTT(f,e)$ is generally not known, this exact minimal simulation time can usually not be determined in practice. However, this is not necessary for practical applications and does usually not yield the preferred simulation time either, as will be discussed next.

Instead of applying this exact value, we can use a practical approximation by replacing the $WCTT(f,e)$ with an upper bound computed by an analytical method or based on maximal values observed in previous simulations, thus creating an adaptive approach to determine a more precise value as new information is gathered. As it is often not desirable to adjust the simulation time for every simulation instance, $NSO(SEND(f))$ can be replaced by the maximum range determined for the node start offsets as done in the experiments.

Now that the theoretically possible minimal simulation time has been defined, its practical relevance needs to be assessed. This lower limit on the simulation time gives us a good way of evaluating the maximum possible parallelization factor for a given simulation budget. However, this theoretical boundary does not consider the added cost of initializing the simulation process, which remains constant independently of the simulation time. The simulation speedup factor of increasingly shorter simulations drops continuously with the reduction in simulation time, as the fraction between computational resources spent on initialization and resources spent on simulation increases. When the initialization portion gains significance in comparison to the time spent on the simulation itself, the overall simulation may no longer be efficient as a non-negligible portion of the simulation duration is spent on initialization alone. This effect can be observed from Figure 4.1.

On the other side of the spectrum, for increasing simulation times, the speedup factor generally tends to drop, too. This can happen due to the technical properties of the simulation software. Concretely, the simulation process generates data and needs to accumulate, evaluate and store results. This can lead to memory management that costs additional resources, thus reducing the speedup factor as work is spent on operations not related to traversing simulation states. As discussed in Chapter 2, the simulation software employed in this work runs in constant memory, and a drop-off of the speedup factor for longer simulation should thus not be expected in a significant capacity.

4.2 Research Questions

The following research questions are defined to derive methods on how the simulation time and node start offsets should be chosen. Their effectiveness is evaluated on an updated set of test cases as presented next. The research questions investigated in this chapter can be formulated as follows:

RQ4. How should the simulation time be set in order to maximize the efficiency of aggregated short simulations?

RQ5. How can we determine suitable node start offsets that maximize the benefits of aggregated short simulations?

RQ6. How well does the simulation aggregation perform on our test cases in comparison to the classical approach when applying the methods derived in RQ4 and RQ5.

These research questions aim to evaluate how well the simulation time determined by the heuristic developed in this chapter works in comparison to the baseline solution presented in the previous chapter. Additionally, the research questions are designed to evaluate the effectiveness of the stratified sampling of the node start offsets in combination with the derived simulation times.

4.2.1 Approach to determine Simulation Time

A pretest is proposed to determine the optimal simulation time t_{sim}, which implicitly determines the average simulation speedup factor of short simulations. The pretest is performed by allocating a certain portion of the simulation budget to evaluating the speedup of simulation times of different lengths. This portion should be selected such that half of it yields a representative simulation speedup factor comparable to longer simulations. This, again, can depend on the specific simulation software, network configuration, and total available simulation budget.

In the context of this work, a portion of one percent of the simulation budget is selected and could generally be applied as a rule of thumb, given the overall budget is appropriately large. A reference speedup factor is then generated by running a simulation for half of the portion we dedicated to determining t_{sim}. The other half is then used to determine the speedup factors of gradually decreasing simulation times, starting by dividing the remaining pretest budget repeatedly by two until the resulting simulation speedup falls below a certain threshold. For instance, a reasonable threshold could be below 50 to 70 percent of the reference value.

It should be noted that the hardware and software environment can have a significant impact on the results of this pretest. Further, the specific simulation parameter choices for the experiments can have an important impact, too.

To illustrate this, a speedup experiment based on randomized node start offsets was conducted, and its results are shown in Figure 4.1. The results shown allow to observe the general behavior of speedups for different simulation times. This experiment was executed on a different machine using a different software environment and differing parameters, compared to the remaining experiments run in this chapter. It can be observed that the simulation times t_{sim} that would be determined according to this plot would be lower compared to the simulation times that were determined for the experiments as presented in Table 4.2.

In comparison, this simulation speedup experiment used randomized node start offsets, which favor speedup in shorter simulations, as the initial traffic peak (as observed in Figure 3.11) is far smaller, thus leading to a higher speedup. It can also be observed from Figure 4.1 that the results can fluctuate to a certain degree, depending on the network characteristics and parameters. Consequently, it may be beneficial to repeat the evaluation of the simulation speedup factors multiple times. They could be evaluated in parallel, for instance, and the simulation time choice could be made on an observed average, particularly if the simulation budget is very large.

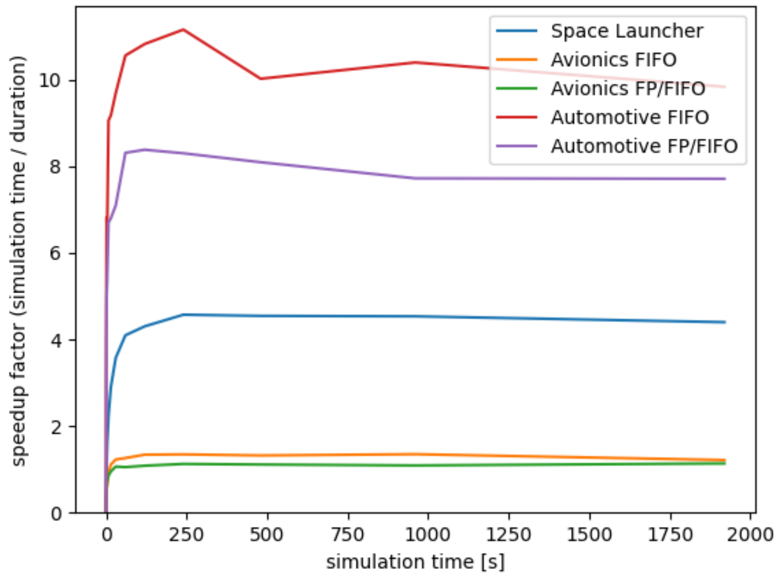


Figure 4.1: **Evolution of simulation speedup factors in relation to the simulation time.**
This experiment shows the behavior of the speedup factor in relation to the simulation time. The reported speedups are evaluated on a single simulation for each simulation time. It can be observed that the speedup factor drastically decreases when the simulation time is minimized. The illustration serves for illustration purposes only and does not directly correlate to the results reported in Table 4.2

The simulation times determined for the evaluation experiments reported below were conducted using synchronized node start offsets for all simulations to create a more conservative choice of simulation times, as the stratified sampling can generate close to synchronized NSO configurations that yield higher traffic and lower simulation speedups. Additionally, the evaluations of the speedup factors were parallelized and chosen based on the average speedup observed.

4.2.2 Stratified Sampling of the Node Start Offsets

As discussed in the previous chapter, the baseline solution to approximating WCTT via simulation traditionally involves running long simulations with synchronized node start offsets. This leads to high interferences in the switches, but can be far from the optimal solution as demonstrated by the unfavorable scenario simulation trace shown in Figure 4.2. In the illustrated simulation trace, packet ”AD2” needs to be released only around $t = 1.04$ms, which would be the node start offset of end-node ”Display1” to generate the illustrated traffic scenario. Further, it was discussed in the previous chapter that the exploration based on clock drifts can be slow and inflexible.

As a consequence, it is desirable to use a larger range to randomly sample the node start offsets from. However, the drawback of increasing this range is that it significantly reduces the chances of generating configurations where the node start offsets for all end-nodes are close to the synchronized case. For instance, to give an intuition of the probabilities involved, if we sample from a $[0.0, 10.0]$ms range, the probability that a node start offset of a node falls within the range of $[0.0, 0.1]$ms is $0.1/10.0 = 0.01$ assuming uniformly distributed sampling. Consequently, the probability that the sampled offsets for all nodes fall within this range exponentially decreases with the number of nodes. For instance, assuming only ten nodes, the chance of generating a configuration where all start offsets for all nodes fall in the desired range of $[0.0, 0.1]$ is $0.01^{10} = 10^{-20}$. This implies that generating a node start offset configuration that is close to the synchronized case is practically impossible when uniformly sampling from a $[0.0, 10.0]$ms range.

As observed in Chapter 3, the aggregation with NSO configurations close to the synchronous case yields high traversal times in general. As a consequence, we still would like to explore the range closer to the synchronous case. At the same time, we would also like to explore the solutions at a further distance from the synchronized case. Essentially, this can be considered an exploration-exploitation dilemma as known from Reinforcement Learning. We want to ”exploit” the knowledge that the synchronous case can yield high traversal times but simultaneously allow the exploration of more diverse node start offset configurations.

A possible solution to integrate this exploration and exploitation in a single sampling method is to equally distribute the samplings to n ranges of different sizes defined by

$$\{[(NSO_{max} - s_i)/2, (NSO_{max} + s_i)/2] \mid i \in \{0 \,..\, n{-}1\}\}$$

where NSO_{max} is the upper limit for the node start offset range, $s_i := NSO_{max} * 10^{-i}$ is the size of the band i around the center of the maximal node start offset range and n is the number of ranges to consider. The parameter n further describes how close to the synchronized case the generated sampling ranges will be, as the resulting ranges become narrower the higher n is. It is suggested to equally distribute the number of samples taken to all ranges, thus $sample\_count/n$ samples will be taken from each range. This sampling approach distributes the efforts to multiple, exponentially growing, and overlapping ranges. This allows the exploitation of narrow search ranges around the synchronized case and simultaneously provides exploration opportunities which can be balanced by the choice of the number of ranges n.

In this dissertation, we chose $n = 5$, and the sample count is determined by the chosen short simulation time and total simulation budget by

$$sample\_count := t_{sim}/t_{total}$$

where t_{total} denotes the total simulation time budget. Determining the maximal node offset range NSO_{max} will be discussed next.

4.2.3 Determining the Maximal Node Start Offset Range

To apply the stratified sampling for the node start offsets as presented in the previous section, a maximal range NSO_{max} needs to be defined. Indeed, there is no technical limit to setting this value but a reasonable range can be motivated. In fact, a suitable value here is the maximal WCTT of all flows. The idea behind this choice is that a flow can only be impacted by other flows during its transmission from sender to receiver, which is bounded by the WCTT. Technically speaking, a flow can be impacted by other flows indirectly via transitivity. However, this effect would be observed as a delay of flows that directly impact the flow of interest and would be indiscernible from a directly impacting flow that is delayed by a certain node start offset. The maximal NSO range can thus be defined as:

$$NSO_{max} := max(WCTT(f, e) \mid \forall f \in F, \forall e \in RECV(f))$$

In practice, as discussed before, the WCTT is generally not known. A possible approximation to this value would again be to use an upper bound derived by an analytical method, if available. Alternatively, if no analytical method that gives a reasonable upper bound can be applied, a simulation can be run to determine the highest observed traversal time. A margin can be added to this value to achieve an actionable approximation for selecting the NSO range and simulation time, or the value could be scaled up by a certain percentage instead, for instance, 30, 50, or 100 percent.

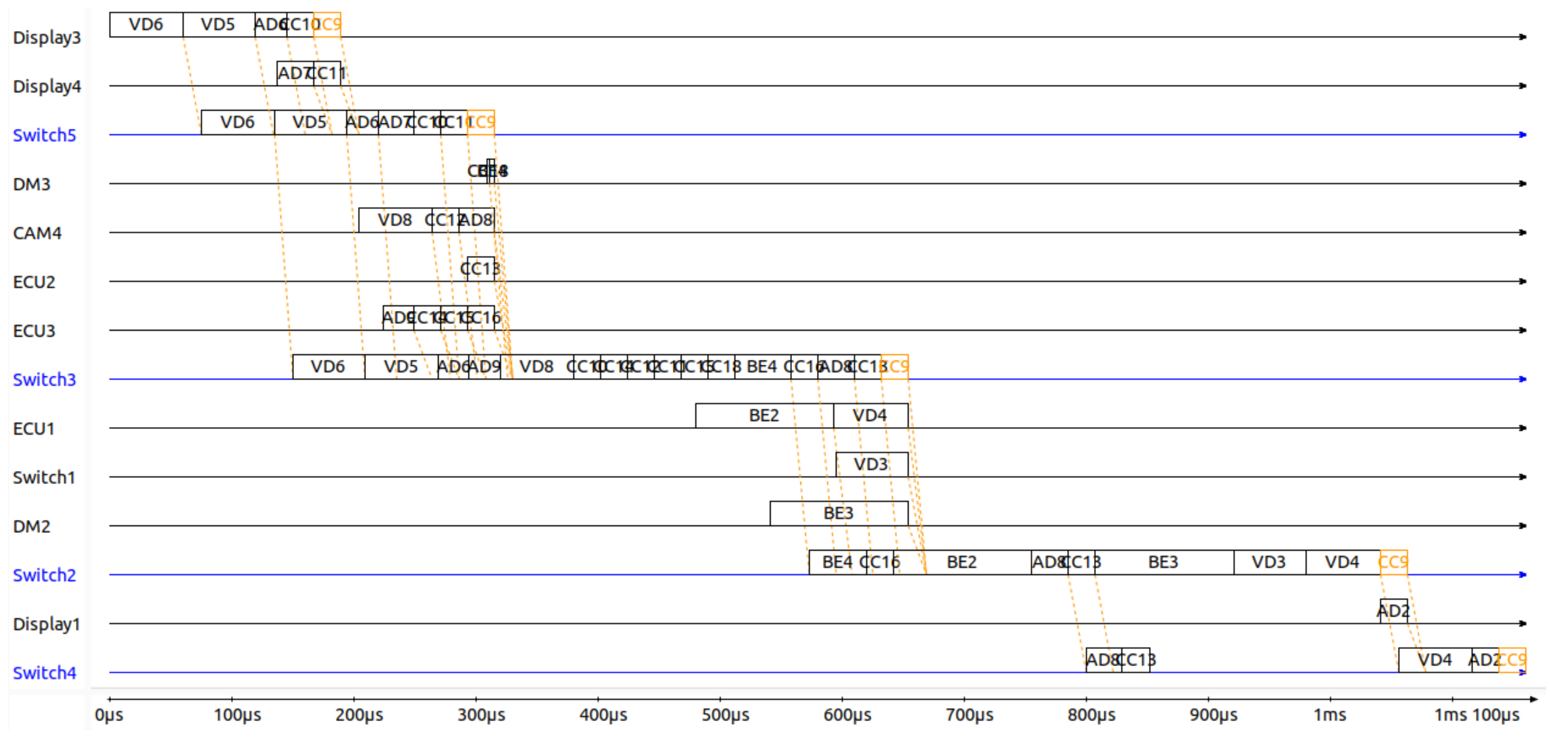


Figure 4.2: **Complex packet trace showing an unfavorable scenario distant from synchronized NSO.**
This packet trace shows a complex unfavorable scenario. It can be seen that the packet for flows BE2, BE3, VD3 and VD4, are scheduled after over $0.45ms$. To achieve this scheduling, the node start offsets of their sender end-nodes, ECU1 and DM2 must be set such that the packets arrive briefly before CC9 reaches Switch2. If these flows were scheduled earlier, they would not contribute to the trace as shown in this scenario, which would effectively reduce the observed traversal time of packet CC9 by (at least) 0.4ms or 40% of its total traversal time observed in this trace. This illustrates why the synchronized NSO case can be far from optimal for generating higher traversal times.

An empirically determined simulation time value could, in fact, be too short to allow the generation of a traversal time for every packet. Luckily, this can be detected easily as the simulation software would not produce an end-to-end delay value for this flow reception. For practical usage, this can be implemented as an adaptive method to approximate the value gradually. As soon as a simulation that does not yield a value for one flow is detected, the simulation time and maximal range value are updated, and the simulation is repeated. This process can be repeated, and the values can be adjusted gradually until the flow yields a traversal time value.

In the stratification experiments run in this chapter, NSO_{max} is chosen based on the highest observed end-to-end delay of all flow receptions during the pretest to determine t_{sim}. As suggested earlier, this observed maximal delay is then scaled up by +50 percent to set NSO_{max}. This maximal range value is used together with a stratification factor of 5 for the experiments using stratification, which is reasonable to attain configurations that are close to the synchronized case for the test cases considered in this chapter.

The NSO_{max} values determined by this heuristic are too large to yield satisfactory results with simple uniform sampling, which was tested in unreported experiments. An alternative pretest was thus performed to determine a more suitable value for the uniform sampling range. This pretest did not yield satisfactory results in all cases, as observed from the results of the avionics configuration with FP/FIFO, which are reported in Table 4.2. This pretest dedicated another fraction of the simulation budget to evaluate the AMTT of different NSO ranges while applying the determined t_{sim}. The ranges determined by this pretest, and applied to the "fixed range" experiments, are reported in the "NSO range" row of the "fixed range" experiment section in Table 4.2.

4.2.4 Evaluation Metric (AMTT)

Evaluating the observed end-to-end delays individually is not practical as the number of flow receptions can grow very large. Consequently, a metric that captures the maximal observed delays for every flow reception across multiple simulation instances is desirable. Such a metric can be defined as

$$AMTT(S) := \sum_{f \in F} \sum_{e \in RECV(f)} max(TT(p_x, e) \mid \forall p_x \in s \in S)$$

where S is a set of simulations s, represented by a set of all packets p_x transmitted during the simulation s, and $TT(p_x, e)$ represents the observed end-to-end delay of a packet p_x for reception e of flow f.

This metric will be referred to as "aggregated maximal traversal times" (AMTT) and represents the sum of the highest observed end-to-end delay per flow reception over a set of simulations, as formalized above. It will be used for evaluating the performance of the experiments conducted in this chapter and in Chapter 5. The maximal observed delays will be determined per flow reception across multiple simulation results and once the per-flow-reception maximum is determined it is summarized to return the AMTT metric. The result figures reported in the following will typically display the AMTT metric on the y-axis to show the magnitude of the end-to-end delays for simulations aggregated up to a certain summed simulation time or simulation duration reported on the x-axis.

4.3 Experiments

The foundations for answering the first two research questions, RQ4 and RQ5, were provided by sections 4.2.1 and 4.2.2, where the methods to determine the simulation time and sampling the node start offsets were derived. For these two research questions, no further experiments are necessary. Instead, the corresponding result sections below will recapitulate the derived methods and provide some additional insights and motivations.

To address research question RQ6 with the aim of evaluating the performance of these methods, two types of experiments have been conducted. One type of experiment evaluates the performance gain of the aggregation approach based on a fixed node start offset range, as done in the previous chapter, while using the simulation times determined by the heuristic presented for RQ4. The other type of experiment applies the same determined simulation times but instead samples the node start offsets from stratified ranges rather than a uniform distribution in a fixed range.

For each experiment, a fixed simulation budget in the form of simulation time is given. This budget is equal for all experiments and is fixed at 100 hours of simulation time. How to dimension the simulation budget depends on the specific use case and is beyond the scope of this work.

In the first step of the experiments, the pretest introduced in Section 4.2.1 is run to determine the speedup factors of different simulation times. As a reminder, this is done by using a certain portion of the simulation budget. Half of this portion is used to determine a reference speedup value comparable to long simulations. For the experiment, one percent of the simulation budget is allocated to the pretest, as suggested before. After determining the speedup factors of different simulation times, the shortest simulation time t_{sim} is chosen such that the resulting speedup factor lies above 90% of the determined reference speedup factor.

The aggregation experiments with NSO uniformly sampled from a fixed range and NSO sampled from stratified ranges are performed based on these determined simulation times. Each experiment was repeated at least 10 times, and the results were averaged over these repetitions to achieve statistically significant and robust results. The reference value reported for the long simulations represents the average over 16 repetitions.

4.3.1 Test Cases

The test cases used for the evaluations in this chapter are mostly equal to the cases presented in Chapter 3.3.2. The topologies remain the same for all configurations, and the traffic configurations remain the same, too, except for one change. For this chapter, the automotive configuration using FP/FIFO+CBS is replaced by a FIFO-only configuration. The motivation to replace this configuration was to enable the usage of the results of the unfavorable scenario analysis as a reference value for comparison and to increase the configuration diversity, as the configurations using FP/FIFO and FP/FIFO+CBS were rather similar.

As they remain unchanged, the topologies depicted in Figures 3.3, 3.4 and 3.5 are still applicable for the current test cases. The updated overview for the configurations applied in this chapter is provided in Table 4.1, showing the automotive configuration using FP/FIFO+CBS replaced by the configuration using FIFO. As for the unchanged configurations, Appendix B provides further details on the added configuration, including the distributions of flow receptions, frame sizes, periods/BAG and deadlines.

Table 4.1: **Characteristics of the updated network configurations for the simulation time and NSO stratification experiments.**
The table summarizes the configurations which are equal to the configurations presented in Chapter 3 in Table 3.1 with one changed configuration. The automotive configuration with "FP/FIFO+CBS" was substituted by a configuration using only FIFO and with TFTP flows removed to increase test case diversity and enable the use of analytical methods to provide a reference value. The table is as presented in publication [54] with adjustments.

Topology	Space Launcher	Avionics		Automotive	
# nodes	18	52		14	
# switches	18	4		5	
# links	24	57		18	
# flows	100	453		58	46
# receptions	985	3214		70	58
QoS	FP/FIFO (4)	FP/FIFO (5)	FIFO	FP/FIFO (4)	FIFO
flow types	99 periodic 1 periodic burst	453 sporadic		35 periodic 11 periodic burst 4 TFTP	46 sporadic

4.3.2 Results

In this section, the experiment results will be presented and analyzed according to the different research questions. As there are no further experiments to be conducted for RQ4 and RQ5, the corresponding sections will provide some context and motivation for the methods developed as a response to either research question, derived in sections 4.2.1 and 4.2.2, respectively.

Results for RQ4: Determining the simulation time

This research question, aimed at developing a heuristic to determine a short simulation time t_{sim} that optimizes the efficiency of the aggregation approach, was implicitly answered in Section 4.2.1. It was discussed before in this work that shorter simulation times offer a greater opportunity for exploration of the simulation state space and increase the probability of observing critical instants. However, reducing the simulation time is subject to certain limitations.

For illustration purposes, an experiment based on the automotive configuration using FP/FIFO is run to demonstrate how increasingly shorter simulations can improve the exploration potential. The experiment performs and aggregates sets of simulations with simulation times of 0.1s, 1.0s, 10.0s, and 100.0s each, for a total simulation budget of one hour each. All node start offsets are uniformly sampled from a fixed range of $[0.0, 20.0]$ms. The results of this experiment are shown in Figure 4.3. It illustrates the beneficial exploration potential of shorter simulations. It is important to emphasize that the experiment compares simulation time, not the actual simulation duration.

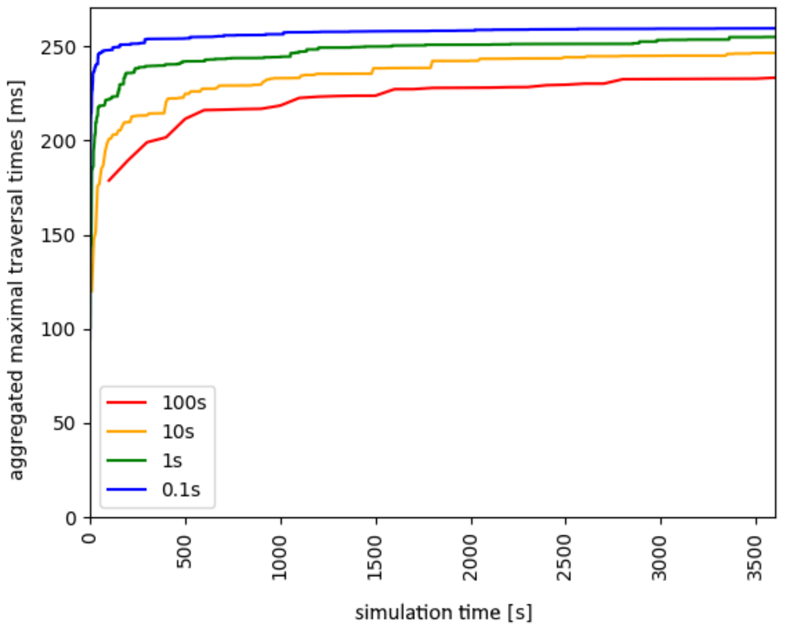


Figure 4.3: **Exploration potential of shorter simulations demonstrated on the automotive configuration with FP/FIFO.**
The figure shows that for aggregations using shorter simulation times (represented by the different curves), the aggregated simulations yield a higher aggregated maximal traversal time (AMTT, y-axis, in ms) for a given simulation time budget (x-axis, in s). Figure as presented in [54].

The results of this short experiment encourage the reduction of the simulation time for short simulations in aggregation. When determining an efficient minimized simulation time, the simulation speedup factor appears to be a limiting factor, as discussed before, which motivated the design of the proposed approach to determine t_{sim}. The proposed approach aims to provide a trade-off between increasing the exploratory potential of the aggregation approach and maintaining a reasonable speedup of the simulations.

Results for RQ5: How to determine suitable node start offsets

This research question aims to improve the sampling of node start offsets by exploiting starting conditions close to the synchronized case while also exploring more complex interaction scenarios using larger offsets. As we have observed in this work so far, the node start offsets, respectively their sampling range, can significantly impact the effectiveness and efficiency of the simulation aggregation approach.

In Figure 4.2 it was observed that the synchronized case does not necessarily generate scenarios that are anywhere close to the worst-case for some flows, while it may be an acceptable approximation of the WCTT for other flows. Thus, it is desirable to allow both, exploration of configurations distant to the synchronized case, while also exploiting the known and understood properties of configurations in close proximity to it.

When uniformly sampling from a fixed sampling range, however, it was demonstrated in a simple example in Section 4.2.2 that the probability of observing a configuration close to the synchronized case quickly becomes very unlikely with increasing maximal node offset range NSO_{max} and increasing node count. The stratified sampling approach presented in the same section is directly designed to address this issue and thus answers this research question.

Results for RQ6: Evaluation of simulation time and stratified node start offsets

In the context of research questions RQ4 and RQ5, methods have been motivated and derived to determine efficient simulation times, and to solve the shortcomings of node start offsets uniformly sampled from a fixed range. Research question RQ6 aims to evaluate these methods based on the updated test cases presented in Section 4.3.1. For each test case, two types of experiments were conducted, as described in this section. The results of these experiments are shown in Figures 4.4, 4.5, 4.6, 4.7, 4.8 and the important information are summarized in Table 4.2.

The results will be discussed subsequently per test case. The long simulations run according to the traditional approach will be denoted as ”single long” in the figures and in the table. The aggregation with NSO sampled from a fixed range will be denoted as ”fixed range”, and the aggregations experiment with NSO sampled using the stratification as presented in Section 4.2.2 are denoted as ”stratified range”. Further, the results of the unfavorable scenario analysis (denoted as ”unfavorable bounds”) and the schedulability analysis (denoted as ”schedulability bounds”) based on Network Calculus will be reported where applicable or otherwise be shown as ”N/A”.

For each experiment, the AMTT metric (see Section 4.2.4) will be reported and for the aggregation experiments (”fixed range” and ”stratified range”) the crossing time (if applicable) will be reported, which marks the simulation time after which the AMTT value of the long simulation (”single long”) was surpassed. For convenience, we additionally report the percentage of by how much the AMTT of long simulation was surpassed (denoted as ”difference to long” in the table).

For the avionics configuration with FIFO traffic management, the following results can be observed in Figure 4.4:

- Uniformly sampled NSOs (”fixed range”) produced an AMTT of 5844.00ms on average, while the result of long simulations (”single long”) was surpassed after only 4050 seconds (about 67.5 minutes) of simulation time. Over the total length of the experiment, the improvement of the AMTT metric is +16.90% over the 4999.07ms of long simulations.

- Sampling from the stratified NSO ranges (”stratified range”) helped improve the AMTT to 6116.37ms, which represents a 22.35% improvement over the long simulation. It surpasses the long simulation AMTT slightly later after 1575 seconds.

- The reference values from unfavorable scenario analysis available for this case is 9376.10, so the stratified experiment is still about 34.7% away from that result.

- The upper-bound provided by schedulability analysis for this case is 9665.98ms and stratified results are thus 36.92% lower.

Overall, a significant improvement can be observed for both determining the NSO via uniform sampling and via stratified sampling. The stratified approach allowed a total average improvement of 22.35% over long simulation and reached an equivalent result after only 0.44% of the total simulation budget, thus representing an aggregation speedup of 228.6.

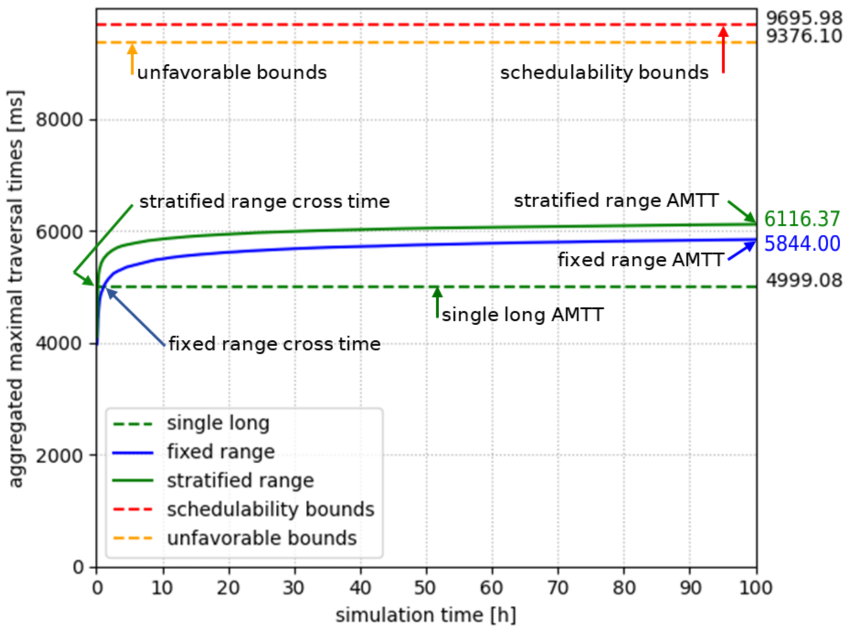


Figure 4.4: **Results for RQ6 - Avionics configuration with FIFO.** The plot shows the evolution of the aggregated maximal traversal times (AMTT, see Section 4.2.4) on the y-axis over the simulation time of the aggregation (x-axis). The simulation aggregation using a stratified node start offset range ("stratified range") is compared to the simulation aggregation with a fixed node start offset range ("fixed range"), and long simulations run according to the baseline solution ("single long"). All variants use a total simulation time budget of 100h. The data of each variant represents the average of at least 10 executions. The long simulation value is plotted as a horizontal line as long simulation does not provide the progression of the traversal times over time. Bounds computed via scheduling analysis and unfavorable scenario analysis are included (where available) for reference. Additionally, on this plot, core information summarized in Table 4.2 is annotated for easy reference. This plot is based on [54] and was updated for clarity.

For the avionics configuration with FP/FIFO, the following results are observed in Figure 4.5:

- Uniformly sampled NSOs produced an AMTT of 3701.25ms on average, while the result of long simulations was not surpassed. Over the total length of the experiment, this represents a loss of about -9.14% over the 4073.57ms of long simulations.
- Sampling from the stratified NSO ranges helped improve the metric to 4660.83ms, which represents an improvement of 14.42% over the long simulation. The stratification approach surpasses the long simulation results after 4500 seconds (75 minutes).
- The unfavorable scenario analysis is not available for this case.

- The upper-bound provided by schedulability analysis for this case is 7011.02ms

The results of the experiment using uniform sampling of the NSO were very bad for this configuration, which may be related to the determined fixed NSO range of 0.05ms, which is extremely narrow. The stratified sampling, however, yielded an important improvement over long simulations. An average improvement of 14.42% was observed, and the equivalent result was achieved after only 1.25% of the simulation time budget, representing an aggregation speedup factor of 80.

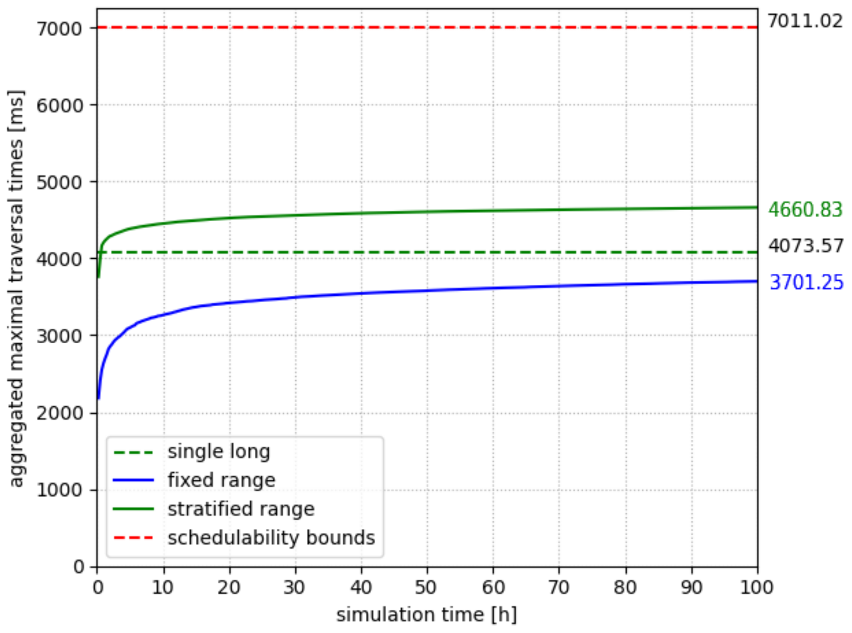


Figure 4.5: **Results for RQ6 - Avionics configuration with FP/FIFO.** The plot shows the evolution of the AMTT (see Section 4.2.4) on the y-axis over the aggregated total simulation time (x-axis) for the experiments using uniform and stratified sampling ranges for the NSO. See Figure 4.4 for further details and annotations.

Figure 4.6 shows the results for the automotive topology with FIFO. The following results are observed:

- Uniform sampling of the NSO (”fixed range”) produces an AMTT of 28.32ms on average. The result of long simulation of 24.41ms is thus surpassed after only 1125 seconds. The overall improvement is +16.02%.

- Sampling from the stratified NSO ranges (”stratified range”) helps to improve the sum to 28.44ms, which represents an additional 16.51% over the long simulation.

The stratification surpasses the long simulation result slightly later after 1575 seconds (26.25 minutes).

- The reference values provided by unfavorable scenario analysis available for this case is 35.85ms, so the stratified experiment is about 20.6% away from that result.
- The upper-bound provided by schedulability analysis for this case is 41.98ms

The improvements of the stratified approach over long simulations are 16.51% and the equivalent result is reached after only 0.43% of the simulation time budget, representing an aggregation speedup of about 228.5.

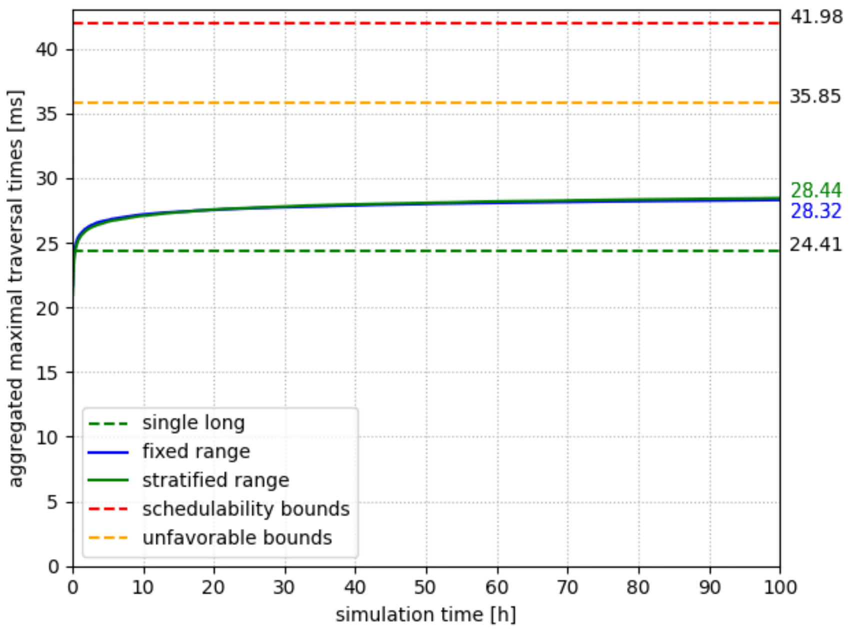


Figure 4.6: **Results for RQ6 - Automotive configuration with FIFO.**
The plot shows the evolution of the AMTT (see Section 4.2.4) on the y-axis over the aggregated total simulation time (x-axis) for the experiments using uniform and stratified sampling ranges for the NSO. See Figure 4.4 for further details and annotations.

For the automotive topology with FP/FIFO, the results are depicted in Figure 4.7:

- Uniformly sampled NSOs ("fixed range") produced only surpasses the long simulation results of 257.18 by only 1.38% and takes 70987 seconds (19.7 hours) of simulation to produce an equivalent result.
- Sampling from the stratified NSO ranges ("stratified range") does not yield a great improvement either and only surpasses long simulation by about 1.73% and surpasses long simulation AMTT slightly earlier after 54956 seconds (15.3 hours) of simulation time.

- The unfavorable scenario analysis is not available for this case.
- The upper-bound provided by schedulability analysis is not available for this case either as best effort flows and TFTP traffic are not supported by the applied analysis. For the other flows, the bound is about 111.58ms.

The results of the experiments for this configuration do not show a significant improvement. The AMTT observed in long simulations is only surpassed by 4.45ms or 1.73% by the stratified experiment. However, the equivalent result of long simulations is reached after only 15.3hours, or 15.3% of the simulation budget, which still represents a speedup of a factor 6.55.

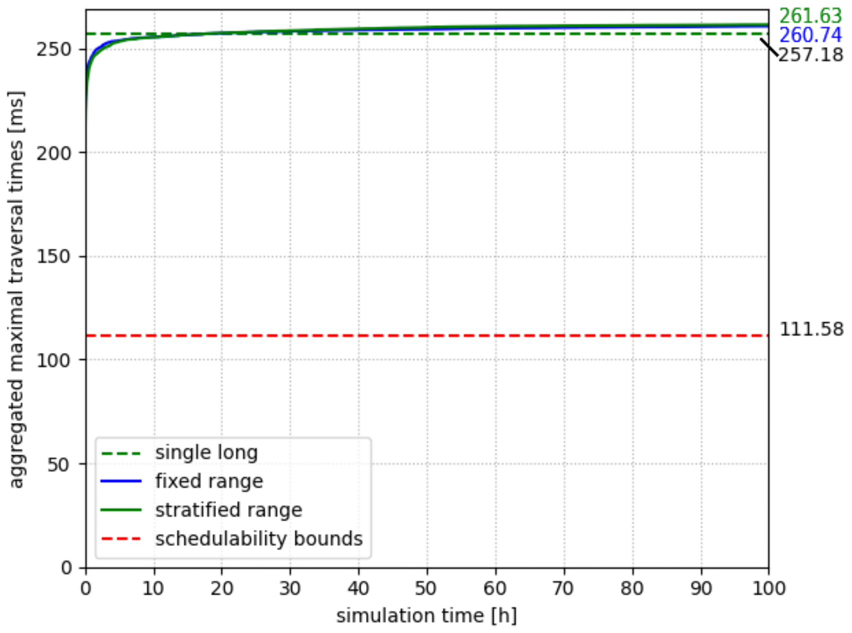


Figure 4.7: **Results for RQ6 - Automotive configuration with FP/FIFO.** The plot shows the evolution of the AMTT (see Section 4.2.4) on the y-axis over the aggregated total simulation time (x-axis) for the experiments using uniform and stratified sampling ranges for the NSO. The displayed schedulability bounds are lower as they exclude TFTP and Best-Effort flows, which are not supported by the applied analysis. See Figure 4.4 for further details and annotations.

The final experiment results to analyze belong to the space launcher topology and are shown in Figure 4.8:

- The long simulations ("single long") give an average sum of 662.6ms, which is surpassed after 22950 seconds (6.375 hours) by the uniformly sampled NSO experiment ("fixed range"). The improvement is about 10.01%

- Sampling from the stratified NSO ranges ("stratified range") improves even further up to 774.0ms, which is an improvement of 16.94% over the long simulation. It also surpasses it much sooner, after only 1350 seconds (22.5 minutes) of simulation time.
- Unfavorable scenarios analysis is once again not available for this case.
- The upper-bound provided by schedulability analysis for this case is 1874.5ms, which is 142% over the result of the stratified sampling experiment.

For this configuration, the improvements are again very significant as the results surpass long simulations after only 1350 seconds or 0.375% of the simulation time budget, this represents an aggregation speedup of a factor $(3600s/h * 100h)/1350s \approx 266$.

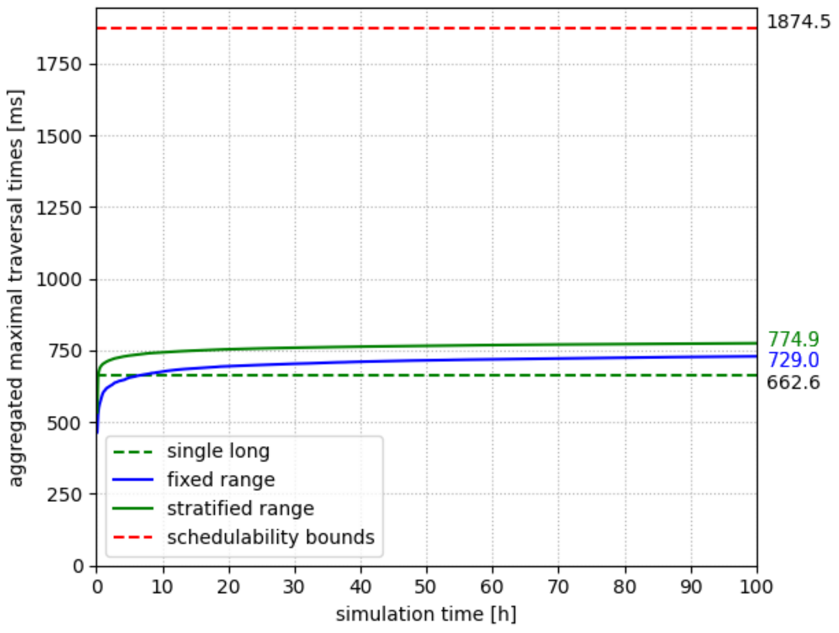


Figure 4.8: **Results for RQ6 - Space Launcher configuration with FP/FIFO.** The plot shows the evolution of the AMTT (see Section 4.2.4) on the y-axis over the aggregated total simulation time (x-axis) for the experiments using uniform and stratified sampling ranges for the NSO. See Figure 4.4 for further details and annotations.

Table 4.2: **Result summary for RQ6 - Experiments with uniform and stratified sampling of NSO.**
The columns represent the different test configurations, identified via topology and applied Quality-of-Service (QoS) mechanism. Rows correspond to the result values as annotated in Figure 4.4. The simulation time t_{sim} was applied for both the "fixed range" and "stratified range" experiments. It was determined using the pretest presented in Section 4.2.1. The "single long" row provides the long simulation experiment results as a reference based on the aggregated maximal traversal times (AMTT) metric presented in Section 4.2.4. The following two blocks in the table represent the experiments using uniform sampling from a fixed range (denoted as "fixed range") and the experiments using stratified sampling (denoted as "stratified range"). The "NSO range" and "NSO_{max}" were determined as explained in Section 4.2.3. The "cross time" denotes the aggregated simulation time when the AMTT of long simulation is exceeded, respectively crossed. The relative difference of AMTT between the long simulation and aggregation experiments is denoted as "diff. to long". The bounds generated by schedulability analysis and unfavorable scenario analysis are provided for reference where available. This table is an extended version of the result table presented in [54].

	Topology	Space	Avionics		Automotive	
	QoS	FP/FIFO	FIFO	FP/FIFO	FIFO	FP/FIFO
t_{sim} [s]		450	450	900	225	56.25
single long	AMTT [ms]	662.6	4999.07	4073.57	24.41	257.18
fixed range	NSO range [ms]	1.48	1.04	2.99	0.05	3.26
	AMTT [ms]	729.0	5844.00	3701.25	28.32	260.74
	diff. to long	+10.01%	+16.90%	-09.14%	+16.02%	+01.38%
	cross time [s]	22950	4050	N/A	1125	70987
stratified range	NSO_{max} [ms]	6.99	4.90	4.92	1.13	15.36
	AMTT [ms]	774.9	6116.37	4660.83	28.44	261.63
	diff. to long	+16.94%	+22.35%	+14.42%	+16.51%	+01.73%
	cross time [s]	1350	1350	4500	1575	54956
unfavorable bounds [ms]		N/A	9376.10	N/A	35.85	N/A
schedulability bounds [ms]		1874.5	9695.98	7011.02	41.98	N/A

4.4 Conclusion and Discussion

It can be concluded that the combination of applying a speedup-maintaining simulation time and the stratification approach yields great benefits over the traditional approach of running long simulations in most cases. Further, the stratified sampling strategy is preferred over uniform sampling from a fixed range as it yields improvements in all cases. It was observed that the equivalent results compared to the baseline solution could be achieved in a fraction of the time, reaching from 15.3% in the worst and 0.375% in the best test scenario. This means that within the same simulation time budget, we could evaluate from 6 to 266 configurations to an equivalent result using the same simulation budget compared to the traditional long simulation approach.

Overall, it can be concluded that the overlapping stratified sampling of the NSO yields a significant improvement at virtually zero added cost besides determining the upper limit NSO_{max} of the NSO range, which can be chosen based on the highest observed end-to-end delay of all flows, as discussed in Section 4.2.3. To maintain constant simulation times for all aggregated simulations, this upper limit was determined by applying a simple scaling of 1.5 to the maximal traversal time observed during the pretest to determine the simulation time t_{sim}, described in Section 4.2.1. In practice, the simulation time (and thus the NSO maximal range) could alternatively be adjusted dynamically throughout the experiment according to the previously observed maximal end-to-end latencies.

As further discussed in Section 4.2.3, determining a reasonable fixed range for the uniform sampling approach is difficult. It can thus be concluded that the stratified sampling approach is generally preferred as it shows equal or better results in all experiments compared to uniform sampling, and the choice of its maximal range NSO_{max} is more practical.

When evaluating many similar variations of configurations whose traffic characteristics do not change significantly, it may be sufficient in practice to run the pretest to determine the simulation time t_{sim} for only a single configuration. The simulation time determined by the heuristic presented in this chapter is usually much larger than the minimal simulation time required to achieve at least one reading of traversal times for every flow. Consequently, some efficiency may be lost in terms of simulation speedup, but the simulation time should still be reasonably dimensioned. In general, if the simulation budget is defined strictly, the simulation aggregation can be stopped as soon as the budget is exhausted. Fluctuations in the simulation speedup can thus be counter-balanced by running additional or fewer simulations to meet the exact simulation budget.

Further, as discussed in the previous chapter, the aggregation approach enables a simple and effective way to parallelize WCTT evaluation via simulation. These parallelization properties are maintained when applying the improvements presented in this chapter, as no significant changes in the approach structure are introduced besides the pretest to determine t_{sim}.

In the following chapter, the increased exploratory potential of reducing the simulation time to the minimum, regardless of the speedup factor, will be explored. Further, the selection of node start offsets will be improved by applying optimization algorithms with an initial population based on the stratification approach presented in this chapter.

Chapter 5
Improving Starting Conditions via Multi-Objective Optimization

In this chapter, the problem of maximizing the observed end-to-end delays will be remodeled as an optimization problem. More specifically, the problem of finding a set of starting conditions that maximize the observed end-to-end delays for each individual flow reception can be represented as a Pareto problem, as discussed in Chapter 3. Such problems are often solved using an algorithm class known as multi- or many-objective optimization algorithms. The optimization technique employed in this chapter is NSGA-II, a simple and popular multi-objective evolutionary algorithm (MOEA).

Another important change that will be investigated in this chapter is the reduction of the simulation time to the minimum to evaluate the exploratory potential of such short simulations. This will cause the simulation speedup factor (see speedup variants in Section 2.4.4) to be reduced substantially, as described in the previous chapter. On the other hand, the reduction in simulation time enables the optimization algorithm to explore the search space further with the given resources and increase the effectiveness of the optimization.

Further, the results of Chapter 4 serve as a foundation for designing a biased initialization for the optimization algorithm to boost the initial starting conditions and, thus, the overall optimization potential of this variation. The effectiveness and efficiency of this approach will be analyzed again based on the five network configurations presented in the previous chapter in Section 4.3.1.

Most of the material presented in this chapter is part of publication [55] and will be referenced accordingly.

5.1 Motivation and Context

As high end-to-end delays are a consequence of the interaction of many flows during their transmission, it is reasonable to hypothesize that multi-objective optimization may be suitable to exploit and improve these interaction patterns of this high-dimensional problem. As discussed and investigated in the previous chapters, the observed end-to-end delays can be influenced by adjusting the scheduling instants of different flows, which can be achieved by adjusting node start offsets and scheduling orders. In this chapter, it is asked whether it is possible to learn interaction patterns incrementally to shift the scheduling instants so that even higher traversal times can be observed. As was illustrated in Figure 3.2, reordering two node start offsets can be sufficient to generate the WCTT of two respective flows that share a certain transmission pattern.

In the previous chapter, sampling from stratified node start offset ranges was further demonstrated to be beneficial as it allows interaction patterns resulting from both synchronized node start offsets and more distant offsets. Consequently, for the sake of optimization, it appears reasonable to also start from such a stratified biased population to raise the general end-to-end latencies observed in the initial population. Additionally, it may enable an improved evolution between different solutions, as they are more diverse and provide patterns that can be recombined to potentially yield even higher traversal times. Without a biased initialization, the optimization algorithm would otherwise need to invest significant resources to discover such synchronized patterns first, which was proven difficult in Section 4.2.2.

5.1.1 Related Prior Work

The most relevant prior work related to this topic has already been discussed in the previous chapters. The general research landscape concerning the dissertation topic was discussed in-depth in Chapter 2. In Chapter 3 and Chapter 4 more specific related work targeting the respective chapter topic was discussed. The related work presented next serves as a brief recap and extends this scope to give an overview of the field of multi-objective optimization.

A similar approach, which focuses on bus systems, was published by Samii, Rafiliu, Eles and Peng in [88] in 2008. Their approach applies and compares various optimization algorithms in order to observe worst-case response times for specific processes of a system. The systems investigated in the paper are based on CAN-Bus and FlexRay, and the approach relies heavily on expert knowledge of the underlying bus systems to prune the search space.

The approach proposed in this chapter is different in many regards, as it considers all flow traversal times simultaneously by exploiting the shared interactions of the many flows. Further, the system models are not comparable since Ethernet and bus systems exhibit vastly different properties and behaviors. Finally, the approach presented in this thesis minimizes the use of expert knowledge and does not prune any of the design variable search space explicitly.

The overlapping stratified sampling technique used to determine node start offsets, as presented in the previous chapter, draws inspiration from the research conducted by Bauer, Scharbarg, and Fraboul in [18] and [19], published in 2010 and 2011, respectively. In their work, unfavorable scenarios are generated based on flow trajectories within Ethernet-based real-time networks. The applicability of this analytical method is limited to sporadic flows, and systems limited to first-in-first-out (FIFO) and fixed priorities (FP/FIFO) QoS mechanisms. The effectiveness of this approach in approximating worst-case latencies was demonstrated in a subsequent study conducted by Boyer et al. in 2012 [24].

This understanding of how scenarios that yield high latencies can be systematically generated played a pivotal role in motivating the development of the stratified approach. Specifically, as observed in Figure 3.2, unfavorable offset configurations can be very similar. These observations served as a basis for the optimization methodology developed in this chapter, as they suggest that optimization strategies could potentially be applied as a generic means of adjusting starting conditions and, thus, the instants when frames are scheduled, to generate unfavorable scenarios in a broader context.

The literature on multi-objective optimization and multi-objective evolutionary algorithms is extensive as these algorithms have been successfully applied to many different fields and applications for decades. Due to their popularity and the variety of problems they can be applied to, a multitude of specialized and improved algorithms were developed. In a survey [98] about MOEAs, published in 2021, Tian et al. give an overview on the topic. They introduce the state-of-the-art algorithms, discuss their respective strengths and weaknesses, and provide insights on the benchmarking landscape for such algorithms.

NSGA-II, the algorithm of choice in this chapter, was developed in 2002 by Deb, Pratap, Agarwal and Meyarivan in [35]. The algorithm represents an improved version of the original NSGA algorithm proposed by Srinivas and Deb in [93]. For completeness, the NSGA-II algorithm will be introduced and discussed in more detail in a separate section. The work [50] by Hort and Sarro from 2021 concludes that the efficiency of NSGA-II can potentially be improved by selecting a smaller offspring size. These findings were applied to select the offspring size in our experiments.

5.1.2 Biased Initialization

As shown in the previous chapter, higher end-to-end delays across the flows can be observed when sampling from stratified ranges rather than a uniform distribution over a fixed range. It is desirable to apply these benefits to the optimization approach as well. Translating the stratification sampling approach to optimization could potentially be done in multiple different ways. One method could be to introduce constraints into the optimization problem to limit the valid ranges of the design variables. This could be used to mimic the effects of the stratified sampling approach presented in the previous chapter. However, introducing such constraints could potentially limit the flexibility of the optimization algorithm and could restrict it from rapidly exploring beyond the narrower ranges.

An alternative method of embedding the stratification approach into the optimization chosen for this work is to generate a biased initial population based on the stratification sampling. In multi-objective optimization research many different sampling methods to generate an initial population have been developed. If no beneficial distribution is known, popular sampling methods include random sampling or Latin Hypercube Sampling (LHS). However, if structural knowledge about the problem is available, biased initialization can be used to speed up the discovery and evolution of the optimization process and can thus improve the convergence towards the optimum.

As the overlapping stratification sampling presented in the previous chapter has shown high effectiveness, it appears useful for generating such a biased initial population. The resulting population is proposed to include the synchronized case, where all offsets are equal. For the remaining population, five overlapping strata of exponentially growing size are generated, as done in Chapter 4. The population of the chosen size is equally distributed across these strata. The choice of the population size depends on the problem complexity induced by the network configuration under investigation and is typically selected relative to the number of objectives, which, in this case, is the number of individual flow receptions.

5.1.3 Pareto Problems and Multi-Objective Optimization

Adjusting starting conditions to observe maximized end-to-end latencies can be represented as a Pareto problem. This means that there is no single optimal solution to the problem, as there are competing objectives in the form of end-to-end delays of different flows, as was demonstrated in Figure 3.2. The objectives are the result of a certain set of design variables, which, in this case, are the starting conditions of the simulation. The fitness function of the optimization problem is given by the network simulation. For these problems, multiple solutions that each maximize individual objectives can be found. Solutions that maximize a certain objective are known as non-dominated or Pareto-optimal solutions. By finding the non-dominated solutions for all objectives, a Pareto front can be defined where each solution is not better or worse than other solutions that belong to this front but merely optimize different objectives.

Usually, Pareto fronts are used to select a solution based on the preferences of the designer or decision-maker of the problem. In the context of maximizing observed end-to-end latencies, however, the goal is to maximize all objectives. Consequently, the goal is to find all Pareto-optimal solutions. The set of all Pareto-optimal solutions would yield all the actual worst-case traversal times for all flow receptions of the network configuration.

Multi-objective optimization (MOO) is usually designed to gradually generate and improve one or multiple Pareto fronts for such problems. Many different MOO algorithms exist that are designed to solve different problems with specific properties. One category of such multi-objective optimization algorithms is multi-objective evolutionary algorithms (MOEA). For the sake of this work, a simple MOEA known as ”non-dominated sorting genetic algorithm 2” or NSGA-II was chosen. It is a well-established algorithm in the field of multi-objective optimization, and due to its simplicity and effectiveness, it is very popular and has been applied successfully to many different research problems. Due to this popularity, many optimized and mature implementations of this algorithm are available in most libraries that support multi-objective optimization. However, the choice of NSGA-II is not the exclusive reasonable choice for the application with simulation aggregation. It was merely chosen for its simplicity and popularity, and other optimization algorithms may, in fact, be more suitable to address the problem and should be investigated in future work. The functioning of the NSGA-II algorithm will be described in more detail next.

5.1.4 NSGA-II

The original ”Non-dominated Sorting Genetic Algorithm”, or NSGA, was developed by Srinivas and Deb in [93]. It was later improved as NSGA-II by Deb et al. in [35] by improving the efficiency, effectiveness and robustness of the algorithm. The original NSGA featured a computational complexity of $O(MN^3)$ where M is the number of objectives and N is the population size. In the second version, this complexity was improved to $O(MN^2)$ by applying an improved sorting strategy. Another improvement over the original NSGA algorithm is the introduction of an elitism method, which selects the best individuals from the current generation and carries their advantages over to the next generation, which improves the convergence towards the optimal Pareto-front. The elitism mechanism considers a crowding-distance and the front rank to carry over the best solutions from the previous generation to the next. The front rank basically describes for each objective a ranking for the quality of a solution. The crowding distance on the other hand is used to retain a possibly high diversity, improving the robustness of the approach as it helps avoiding local maxima.

The NSGA-II algorithm starts by generating an initial population, based on a selected initialization heuristic. This population’s fitness is then evaluated and sorted according to the non-dominated sorting criterion and the crowding-distance is computed. After these initialization steps, until a certain termination criterion is met, the algorithm iterates through 5 different main steps. First, offsprings are generated using the standard genetic algorithm, then the offspring fitness is evaluated, the non-dominated sorting is updated including the offsprings, and finally their crowing-distance is computed and the elitism is performed to retain the best individuals of the highest ranked Pareto fronts according to the population size.

Various termination criteria can be used, such as the number of fitness evaluations, the number of generations, or the execution time. In the context of this work, the number of evaluations is used as a termination criterion as it is directly correlated to the simulation budget as a limiting factor, due to the constant simulation time applied for all simulations.

The standard genetic algorithm itself is composed of three different steps again, tournament selection, crossover and mutation. There are different variations for each of these steps that are independent of the NSGA-II algorithm and can be specified upon initialization of the algorithm. The role of the tournament selection is to select preferred individuals to form offsprings by applying some variation of cross-over. The cross-over is used to recombine the select parent solutions into a new solution that retains certain features of either of its parents. Finally, some mutation operator appropriate to the problem is applied to introduce feature modifications with respect to the parents of the offspring and allow it to obtain new unique properties that may not have been observed in the population before.

Relations between the exploration-exploitation dilemma that was discussed before in this thesis and the genetic algorithm can be made. The crossover can be considered an exploitation strategy, as it retains the features of solutions that are known to perform well, thus exploiting these features. On the other hand, mutation can be considered an exploration strategy, as it allows the modification of features that have proven effective before and thus enables the exploration and discovery of new areas of that feature space.

5.2 Research Questions

Three research questions have been devised to evaluate the effectiveness of the minimized simulation time and the optimization approach. They investigate the effectiveness of the minimized simulation time in isolation, and in combination with the optimization of node start offsets with and without frame scheduling order. Stratification sampling is applied to determine the node start offsets in all cases. Finally, the overhead cost introduced by the optimization and the shortening of the simulation time is investigated. The following research questions are investigated:

RQ7. Can we leverage the exploratory power of shorter simulation by minimizing the simulation time beyond a threshold that would maintain the simulation speedup factor?

RQ8. Can multi-objective Pareto optimization algorithms improve the exploration of the node start offset search space in the approach of aggregating short simulations for approximating worst-case traversal times via simulation?

RQ9. What is the overhead cost of performing the optimization compared to resources invested in simulations?

The research questions and results presented in this chapter were developed in the context of the related paper [55].

5.2.1 Formulating WCTT Approximation as an Optimization Problem

The task of maximizing the observed traversal times by adjusting node start offsets and flow scheduling order can be easily modeled as a Pareto multi-objective optimization problem. To do so, the node start offsets and, optionally, the frame scheduling order via frame offsets, are used as the design space parameters, while the end-to-end latency for each flow reception defines a separate objective. As a reminder, a flow can have multiple receptions in the case of multi-case flows and refers to each individual node that receives a copy of a packet emitted by a flow.

The limitations of the design space parameters are set in the range from zero to the maximal NSO limit NSO_{max} as described in Section 4.2.3. For the frame offset parameters, the valid range is set from zero to the number of flows in nanoseconds. This guarantees that the chosen frame offsets cannot significantly impact the traffic characteristics.

To improve the starting population diversity and, thus, the optimization potential, a biased initial population is drawn according to the stratification sampling as described in Section 5.1.2. When flow scheduling order is optimized in addition, the flows are simply enumerated, yielding integer nanosecond values, which are randomly permutated for each generated individual of the biased population.

5.2.2 Reordering Flow Scheduling Order via Frame Offsets

While the ordering of the flows that are scheduled at the same time was previously steered indirectly via the simulation random seed, it can be desirable to control and optimize this order explicitly. As has been observed from the packet trace generated by the unfavorable scenario analysis that was shown in Figure 4.2, the sending order of packets can be essential for observing unfavorable scenarios resulting in higher traversal times. At the time of writing, the simulation software of choice does not offer an explicit way to define the packet scheduling order of simultaneously scheduled packets on the same node and is exclusively steered by the random seed. We can, however, make creative use of frame offsets as introduced in Section 2.2 to explicitly steer the frame scheduling order. This feature allows the creation of a scheduling gap between packets that belong to different flows originating from the same end-node. In the general case, a system naturally has implicit frame offsets because the packets are generated by certain tasks running on the end-nodes. Unless those tasks are run in parallel, they usually define the sending order, which can be considered random in real systems to a certain extent. Task scheduling is a complex topic of its own and beyond the scope of this dissertation.

As this work aims to remain as generally applicable as reasonably possible, no specific order for the flow scheduling is assumed and can be considered a free variable. Experiments were conducted to investigate the potential of optimizing the flow scheduling order in addition to the node start offsets. As it is important to avoid significantly impacting the traffic characteristics of the configuration, the frame offsets are chosen at the scale of nanoseconds. The effects of such small frame offsets are, in the context of the applied test cases, insignificant compared to the effects of other variables like switching delays or clock drifts.

In some special cases, a specific frame scheduling order for flows originating from the same node may be known. Given such a case, this knowledge can be integrated into the optimization approach in the form of constraints and can help to vastly reduce the possible search space, effectively reducing the complexity of the optimization problem. However, such an ordering is not assumed or given in the context of this dissertation.

5.3 Experiments

Various experiments have been conducted to evaluate the core effectiveness of minimized simulation time on its own and in combination with optimization of node start offsets with and without frame offsets, as proposed in this chapter. The following experiment variations have been conducted for every test case configuration:

- Uniformly sampled NSO (no optimization) with stratified ranges and minimized simulation time (denoted as ”random minimized” in figures and tables).
- Optimizing only NSOs with a biased initial population using stratified sampling and minimized simulation time (denoted as ”opti. NSO”).
- Optimizing NSOs and FOs with a biased initial population using stratified sampling and minimized simulation time (denoted as ”opti. NSO+FO”).

Further, the data of the long simulation experiments and of the short simulation experiments using stratified sampling from the previous chapter have been reused as a reference. The results of the long simulations will be denoted as ”single long” and the results of the short aggregations with stratified node start offsets are denoted as ”random short”.

5.3.1 Test Cases

The exact same configurations as presented in Section 4.3.1 are used as test cases to perform the experiments. These test cases are suitable as they are diverse and of different complexities, with the avionics configurations amounting to as many as 3214 objectives, each represented by every individual flow reception. For these configurations, the design variable space has 52 dimensions for the case where no frame offsets are optimized and as many as $453 \times FO + 52 \times NSO = 505$ dimensions for the experiments that additionally apply frame offsets for controlling the frame sending orders.

Reusing the same test cases as for the previous chapter further allows us to compare the data of optimization-related experiments to the data gathered in the context of the previous chapter, giving the reader an understanding of the relative improvements that optimization can yield.

5.3.2 Experiment Parameterization

Each experiment was run for a total sum of two hours of simulation time, and all experiments were repeated 10 times. For the experiments with minimized simulation time, clock drifts were set to zero to remove the effect of exploration via clock drifts with the intention of creating a more direct relation between design variables and fitness function value. This effect should be minimal anyway because the extremely short simulation time significantly reduces the effects of clock drifts on the observed results.

A new random seed was applied for every simulation instance. The simulation time and maximal node start offset range for each configuration were empirically determined based on previous observations. The simulation times were chosen as small as possible to allow the experiments to focus on the exploratory power of minimized simulation times.

As discussed in the previous chapter, the maximal node start offset range and the minimal simulation time can be set according to the WCTT, or an observed (and upscaled) lower or upper bound.

5.3.3 Results

In this section, the results of the experiments will be summarized and discussed. The provided illustrations for RQ7 and RQ8 were generated for the related publication [55], while some figures had to be omitted from the paper for brevity. All relevant results are therefore summarized again in Table 5.1, which was also part of the publication. Table 5.2 provides the overhead experiment results for RQ9 and was also part of the paper.

Results for RQ7: Minimized simulation time without optimization

This research question investigates the effects of minimizing the simulation time with stratified node start offsets without applying any optimizations yet. We can observe the following results, as shown in Figures 5.1, 5.2, 5.3, 5.4 and 5.5, and summarized in Table 5.1 under the experiment identifier ”random minimized”:

- An improvement over long simulation (denoted as ”single long”) for the automotive configuration with FIFO traffic of +29.37% was observed for only an increased simulation duration cost of +6.81%.

- A moderate improvement over long simulation for the automotive configuration with FP/FIFO of +1.23% was observed for a 7.5 times shorter overall simulation duration.

- In the case of the space launcher topology, an improvement over long simulations of +19.61% was observed for an increased total simulation duration of +67.40%.
- In the case of the avionics topology with FIFO, an average improvement of +25.82% could be observed for a +1.76% longer overall simulation duration.
- For the avionics topology with FP/FIFO, an average improvement of +17.35% over long simulation was observed, at an additionally decreased overall simulation duration of about -3.9%.

Overall, the observed average improvements in the aggregated maximal traversal times metric (AMTT) is substantial.

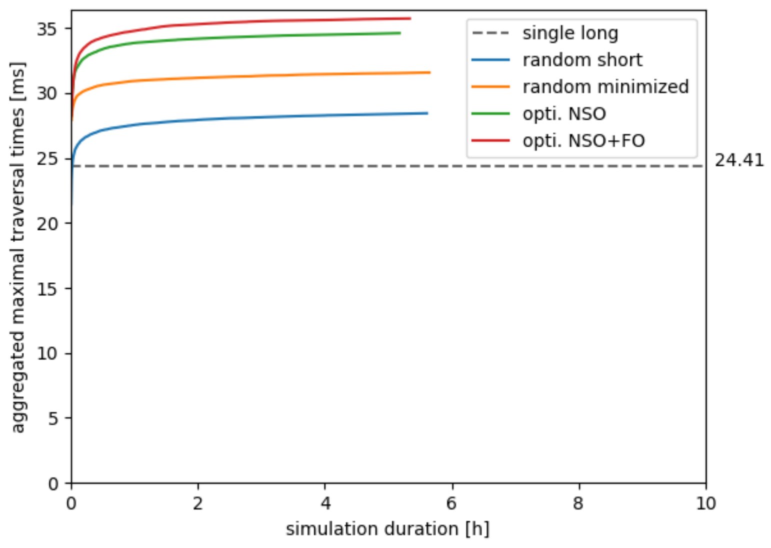


Figure 5.1: **Results for RQ7 and RQ8 - Automotive topology with FIFO.** The y-axis shows the "aggregated maximal traversal times" metric (AMTT, see Section 4.2.4) over the simulation duration of the experiments (up to 10 hours) on the x-axis. The results are shown for the experiments with minimized simulation time (denoted as "random minimized"), and optimization of node start offsets ("opti. NSO") and, optionally, flow scheduling order ("opti. NSO+FO"). The result of long simulations ("single long") and aggregation with stratified NSO (denoted as "random short") from Chapter 4 are included for reference.

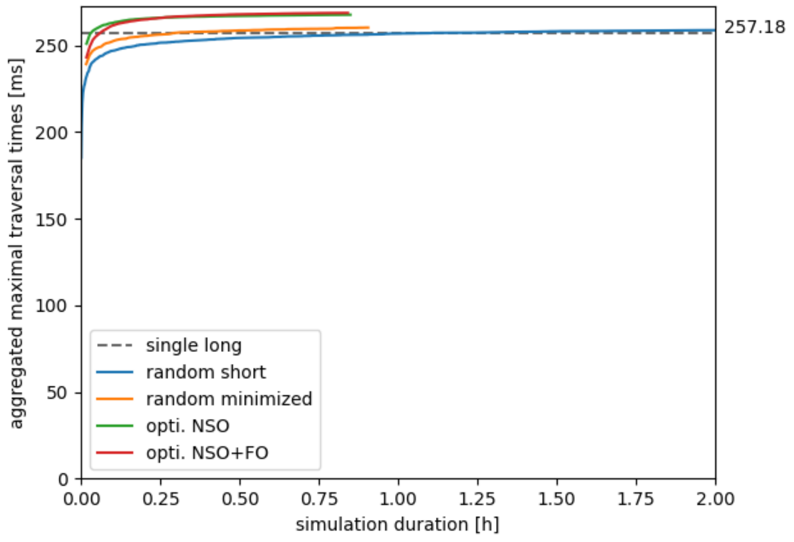


Figure 5.2: **Results for RQ7 and RQ8 - Automotive topology with FP/FIFO.** The y-axis shows the AMTT metric over the simulation duration of the experiments (up to 2 hours) on the x-axis. Further details in Figure 5.1 caption.

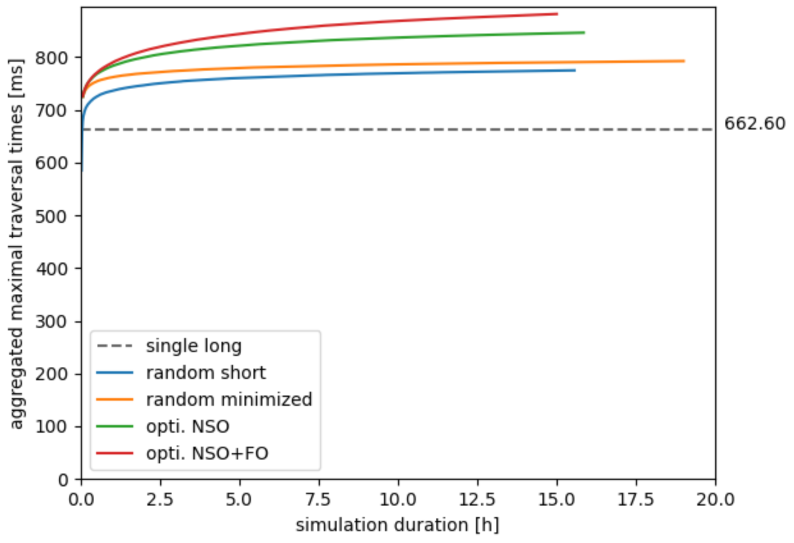


Figure 5.3: **Results for RQ7 and RQ8 - Space Launcher topology with FP/FIFO.** The y-axis shows the AMTT metric over the simulation duration of the experiments (up to 20 hours) on the x-axis. Further details in Figure 5.1 caption. This Figure is presented in publication [55].

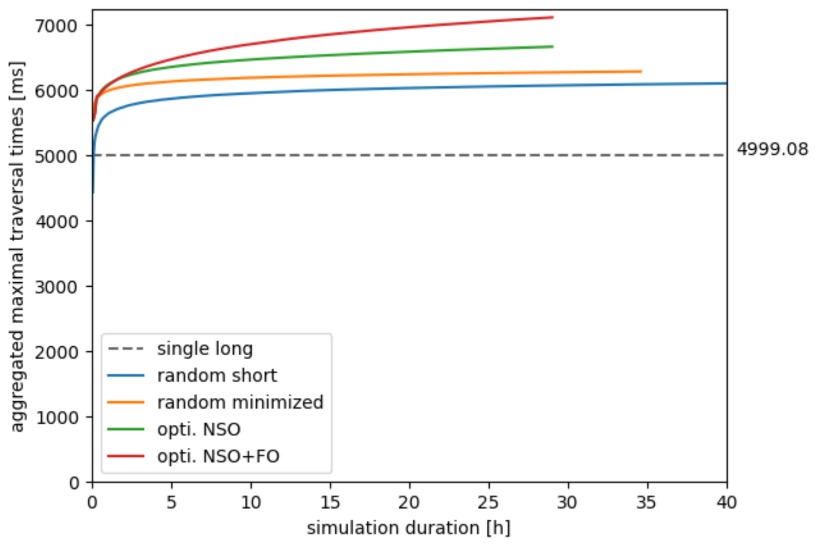


Figure 5.4: **Results for RQ7 and RQ8 - Avionics topology with FIFO.** The y-axis shows the AMTT metric over the simulation duration of the experiments (up to 40 hours) on the x-axis. Further details in Figure 5.1 caption.

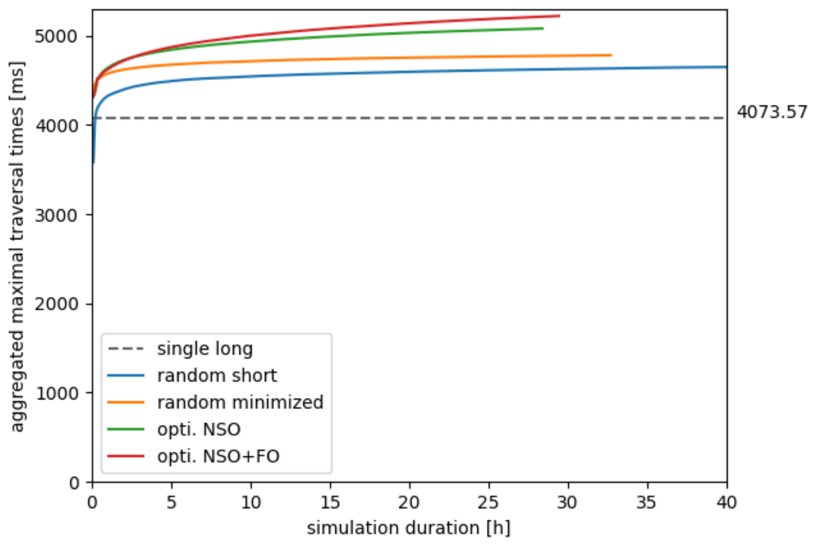


Figure 5.5: **Results for RQ7 and RQ8 - Avionics topology with FP/FIFO.** The y-axis shows the AMTT metric over the simulation duration of the experiments (up to 40 hours) on the x-axis. Further details in Figure 5.1 caption.

Results for RQ8: Optimizing NSOs and optionally FOs

This research question aims to analyze the benefits of optimizing the node start offsets, with and without optimizing the flow sending order. Optimizing the flow sending order is controlled by optimizing minimal frame offsets in the nanosecond range that do not measurably impact the traffic characteristics. In this research question, the overhead of optimization is not considered and will be addressed by RQ9 separately.

The results are included in Figures 5.1, 5.2, 5.3, 5.4 and 5.5, and summarized in Table 5.1. The results are summarized in the table under the identifiers ”opti. NSO” for the optimization of node start offsets only and ”opti. NSO+FO” for the additional optimized flow scheduling orders.

The results can be summarized as follows:

- An improvement over long simulation (denoted as ”single long”) for the automotive configuration with FIFO of +41.79% was observed for a reduced simulation duration of -2.1% when optimizing only NSOs. Optimizing scheduling order in addition, improved by another +4.63%.
- A moderate improvement over long simulation for the automotive configuration with FP/FIFO of +4.12% was observed for an eight times shorter overall simulation duration. Optimizing flow scheduling order in addition, improved by another +0.41%.
- When optimizing only NSOs, in the case of the space launcher topology, an improvement over long simulation of +27.73% was observed for an increased total simulation duration of +40.09%. Optimizing scheduling order in addition, improved by another +5.35%.
- In the case of the avionics topology with FIFO, optimizing NSOs only yielded an average improvement of +33.43% for an additionally -14.7% shorter overall simulation duration. Optimizing frame scheduling order in addition, improved by another +8.97%.
- For the avionics topology with FP/FIFO, an average improvement of +24.70% over long simulation was observed, at an additionally decreased overall simulation duration of about -16.5%. Optimizing the frame scheduling order in addition, improved by another +3.45%.

We observe a consistent improvement at virtually no added cost in simulation duration when optimizing the flow scheduling order in addition to node start offsets. Overall, the optimization (”opti. NSO” and ”opti. NSO+FO”) yields better results than randomized stratified sampling (”random short” and ”random minimized”). It should be noted that the optimization itself comes at an added cost that was not considered in this research question and will be evaluated separately by the next one.

Table 5.1: **Experiment result summary for RQ7 and RQ8.** The table summarizes the results shown in Figures 5.1, 5.2, 5.3, 5.4 and 5.5. The reported speedup is the simulation speedup factor. The experiment annotations are explained in the caption of Figure 5.1. Table as presented in [55].

Configuration	experiment	ST[a]	EST[b][h]	SD[a][h]	speedup	AMTT[c][ms]	wrt. single long[d]
	single long	100 h	100.0	5.29	18.869	24.41	
	random short	225 s	100.0	5.61	17.821	28.44	+16.51%
Automotive FIFO	random minimized	8 ms	2.0	5.65	0.354	31.58	+29.37%
	opti. NSO	8 ms	2.0	5.18	0.386	34.61	+41.79%
	opti. NSO+FO	8 ms	2.0	5.34	0.375	35.74	+46.42%
	single long	100 h	100.0	6.83	14.655	257.18	
	random short	56.3 s	100.0	7.88	12.682	261.62	+1.73%
Autotive FP/FIFO	random minimized	30 ms	2.0	0.91	2.197	260.35	+1.23%
	opti. NSO	30 ms	2.0	0.85	2.353	267.78	+4.12%
	opti. NSO+FO	30 ms	2.0	0.84	2.380	268.82	+4.53%
	single long	100 h	100.0	11.4	8.809	662.60	
	random short	450 s	100.0	15.6	6.425	774.89	+16.95%
Space Launcher	random minimized	18 ms	2.0	19.0	0.105	792.56	+19.61%
	opti. NSO	18 ms	2.0	15.9	0.126	846.37	+27.73%
	opti. NSO+FO	18 ms	2.0	15.0	0.133	881.79	+33.08%
	single long	100 h	100.0	34.0	2.942	4999.08	
	random short	450 s	100.0	43.9	2.277	6116.37	+22.35%
Avionics FIFO	random minimized	14 ms	2.0	34.6	0.058	6289.82	+25.82%
	opti. NSO	14 ms	2.0	29.0	0.069	6670.35	+33.43%
	opti. NSO+FO	14 ms	2.0	29.0	0.069	7118.70	+42.40%
	single long	100 h	100.0	34.0	2.940	4073.57	
	random short	900 s	100.0	46.9	2.130	4660.83	+14.42%
Avionics FP/FIFO	random minimized	16 ms	2.0	32.7	0.061	4780.18	+17.35%
	opti. NSO	16 ms	2.0	28.4	0.070	5079.91	+24.70%
	opti. NSO+FO	16 ms	2.0	28.4	0.070	5220.15	+28.15%

[a] ST: simulation time t_{sim} (time elapsed inside the simulation); SD: simulation duration (wall-clock time)
[b] EST: total experiment simulation time budget
[c] AMTT: ”aggregated maximum traversal time” metric (see Section 4.2.4)
[d] wrt. single long: performance in comparison to the ”single long” reference simulation; higher is better

Results for RQ9: Analyzing Optimization Overhead Cost

This research question evaluates the additional overhead cost of the optimization algorithm and efficiency losses caused by diminished speedup due to the reduced simulation time. The results of this evaluation are summarized in Table 5.2. The reported effective parallel simulation duration (denoted as ”PSD” in Table 5.2) is computed by considering the total simulation duration (denoted as ”SD”) spent on generating the simulation results and is then divided by 12, which was the parallelization factor applied when running the simulations. By doing this, the returned value is comparable to the total experiment duration (denoted as ”ED”) which was tracked from start to finish of an experiment. Thus, dividing the sum of simulation durations by the parallelization factor gives us (approximately) the linear portion of the wall-clock time that was spent to run the simulation. Since there is some minimal synchronization losses from the parallelization, this impact is attributed to the optimization algorithm and the values reported thus give a slightly pessimistic evaluation.

Table 5.2: **Optimization overhead evaluation results for RQ9.**
Table similarly presented in [55].

Configuration	experiment	ED[a][h]	PSD[b][h]	SD[c][h]	overhead[d]
Automotive FIFO	opti. NSO	0.706	0.432	5.18	+63.55%
	opti. NSO+FO	0.729	0.445	5.34	+63.82%
Automotive FP/FIFO	opti. NSO	0.093	0.071	0.85	+31.29%
	opti. NSO+FO	0.092	0.070	0.84	+31.43%
Space Launcher	opti. NSO	1.903	1.325	15.9	+43.62%
	opti. NSO+FO	1.830	1.250	15.0	+46.40%
Avionics FIFO	opti. NSO	8.121	2.418	29.0	+235.93%
	opti. NSO+FO	7.983	2.418	29.0	+230.22%
Avionics FP/FIFO	opti. NSO	7.364	2.367	28.4	+211.15%
	opti. NSO+FO	7.255	2.367	28.4	+206.55%

[a] ED: experiment duration
[b] PSD: parallel simulation duration, time to run in parallel on 12 cores
[b] SD: simulation duration, time to run all accumulated simulation sequentially
[d] overhead: time spent on optimization rather than simulation := $(ED - PSD)/PSD$

For the automotive and space launcher topologies, moderate but significant optimization overheads of 31.29% to 63.82% are observed. In the case of the avionics configurations, the overhead is much more extensive and more than two-thirds of the time is spent on optimization alone, representing an overhead of up to 235.93%. This observation is not surprising as the NSGA-II algorithm’s complexity grows linearly with the number of objectives and quadratically with the population size, which is typically chosen in function of the number of objectives, as discussed above.

It can be observed further that the overhead cost is not larger when additionally optimizing the frame offsets. This may seem surprising at first, but it makes sense as the frame offset increases the number of design variables while the number of objectives remains unchanged, which is one of the main driving factors behind the complexity of the NSGA-II algorithm.

5.4 Conclusion and Discussion

Regarding RQ7, it can be concluded that minimizing the simulation time is generally beneficial despite the additional cost of a decreased simulation speedup factor. In all cases, higher total traversal times could be observed. Additionally, in some cases, the total simulation duration was even shorter compared to long simulation (”single long”) or aggregation of short simulations in the seconds range (”random short”).

In the context of RQ8, it can be consistently observed across the different experiments that optimizing frame scheduling order (by means of frame offsets) in addition to node start offsets is very beneficial at a negligible or no additional cost over only optimizing node start offsets. It can thus be concluded that optimizing the frame scheduling order for the sake of observing higher traversal times is superior to randomly exploring scheduling orders via the unpredictable effects of the random seed.

Unfortunately, we have also observed that the impact of the reduced speedup factor of shortening the simulation time is very dramatic and has dropped by a factor of up to 83.9 from long to minimized simulation time for the space launcher topology.

Another notable thing to observe is that the optimization approach on the automotive case with FIFO traffic achieved an average observed aggregated maximal traversal time (AMTT) of 35.74, which is very close to the 35.85 (+0.11ms) generated by unfavorable scenario analysis, known to produce rather tight bounds on the actual WCTT.

It needs to be highlighted that the experiment simulation time budget *EST* applied for the aggregations was two hours compared to 100 hours for the reference data (”single long” and ”random short”). As all optimizations exceeded the AMTT bounds observed for the reference long simulation, we can conclude an aggregation speedup factor of at least 50 (see Section 2.4.4). The precise crossing point was not explicitly investigated in this chapter but seems to follow similar trends to the observations in Chapter 4, as seen in the figures. This means that, approximately, the actual aggregation speedups in this chapter were about 50 times higher due to the different simulation time budget magnitudes. Unfortunately, the benefits of these increased aggregation speedup factors were counterbalanced by the reduction in simulation speedup, which was the reason for reporting the results in terms of simulation duration rather than simulation time. This, however, provides strong motivation for developing simulation software that is optimized for running very short simulations.

Regarding RQ9, it can be further concluded that the NSGA-II algorithm does not scale well with the increasing complexity of the configurations, as was observed in Table 5.2. For the avionics configuration, the overwhelming majority of the time (more than two-thirds) was spent on optimization rather than simulations. For other configurations, the optimization overhead was significant but still within reasonable margins, considering the improvements in the results.

As discussed before, NSGA-II represents a popular choice for multi-objective optimization in general but may not be the best-suited choice for the aggregation approach, especially regarding scaling to more complex configurations. Further, the loss in efficiency in the form of a reduced simulation speedup, caused by the minimization of the simulation time, is currently a relevant limiting factor to the efficiency of the aggregation approach with optimization. Running extremely short simulations is not a typical use case considered in the development of network simulation software today, thus available simulators are not optimized for it. There may, however, be technical solutions to reducing the simulation initialization cost for running many short simulations of the same network configuration, thus significantly reducing the impact of this currently limiting factor.

Overall, we can conclude that optimization is able to exploit hidden structural interference patterns in flows as the maximally observed end-to-end latencies exceeded the values observed from aggregations without optimizing starting conditions and long simulations in all cases by a significant margin for a comparable simulation duration and drastically reduced simulation time. Despite the additional overhead introduced by the minimized simulation time and optimization, it is observed that the simulation using optimization of node start offsets and frame scheduling order outperforms the previously investigated variants on all considered use cases. It should further be taken into account, that the closer the actual WCTT is approached, the more unlikely it becomes to observe higher traversal times, typically requiring exponentially more effort when relying on the traditional approach.

Parallellization Potential of the Optimization Approach

The optimization can be parallelized to a certain degree and on different levels. One way to parallelize the optimization process is to parallelize the evaluation of the fitness function, namely the simulation, of different individuals from the population. This would be equivalent to how parallelization of the simulations would be realized for the uniform sampling approach. All individuals, once generated by the genetic algorithm, are independent of each other. This means that the simulations to evaluate the fitness of individuals of a generation can be run in parallel up to the degree of the size of the offsprings that are generated at every generation. Compared to running the random search, this means that the potential for parallelization is reduced, as new solutions are generated in generations and are not known upon the initialization of the algorithm.

A different but complementary way to parallelize the optimization is to apply meta-heuristic or meta-optimization strategies. One type of meta-heuristic optimization applicable to this case is known as ”island models”, which belong to the category of parallel evolutionary algorithms.

Island models work on the principle of evolving independent populations on different islands in parallel but regularly exchange or share some of their individuals across islands. This helps to retain diversity while at the same time allowing for an effective co-evolution, often resulting in a faster convergence towards the optimum.

The parallelism gained by such island models can be combined with parallelizing the evaluation of the fitness of the offsprings, thus multiplying the overall parallelism of both approaches. This strategy could be beneficial when a highly parallel infrastructure is available, such as cloud-computing or high-performance-computing (HPCs), and the problems to be solved are of higher complexity. An approach to realize such a parallelization, using an island model in combination with NSGA-II, is described in [72] by Märtens and Izzo.

Chapter 6
Conclusion and Perspectives

This chapter will summarize the overall contributions and conclusions. The different parts will be integrated to form a complete picture and overall conclusion of the work presented in this dissertation. The chapter will then close by presenting several possible future research directions based on the findings and open questions that remained unanswered.

6.1 Synopsis of Contributions

In this thesis, a novel paradigm to evaluate worst-case traversal times via simulation was developed. Crucial parameters, such as simulation time, node start offsets, clock drifts, and frame scheduling order were investigated to maximize the benefits of the approach.

Concretely, a heuristic to set the simulation time is proposed that roughly preserves a simulation speedup factor comparable to long simulations. The process to determine this simulation time is run by dedicating a certain percentage of the overall simulation budget to determine a suitable simulation time for short simulations. Half of this budget, half a percent of the overall budget, for instance, is used to determine a reference speedup. An equal portion is then split up to run simulations of increasingly shorter simulation time and evaluate their corresponding speedup factor. The simulation time is then chosen as soon as the resulting speedup factor drops below a fraction relative to the reference speedup, for instance, 0.9. This approach allows subdividing long simulations into shorter ones without a significant loss of computation efficiency, as the gained exploratory power from aggregated simulations typically outweighs the added overhead significantly, as observed in this work.

Another important contribution to improving the efficiency and effectiveness of the aggregation approach is to sample the node start offsets from multiple overlapping and exponentially growing stratified ranges rather than a single fixed range. This yields the advantage of providing a good balance between the exploration of larger offset ranges and the exploitation of the narrow ranges close to the synchronous case, which have proven to yield observations of reasonably high but often not maximal end-to-end delays .

The final contribution of this thesis is the aggregation variant using multi-objective optimization to determine the starting conditions for short simulations. For the experiments a popular and well-established algorithm from the domain of multi-objective optimization was employed, namely NSGA-II. The thesis has demonstrated how the problem can be modeled as a multi-objective Pareto problem and be solved by applying this off-the-shelf optimization algorithm. In addition to optimizing the node start offsets, it was shown that optimizing the frame scheduling order is generally superior. The frame scheduling order was optimized by applying minimal frame offsets in the nanosecond range to avoid significantly impacting the traffic characteristics.

As a result, with equal resources, the optimization approach managed to achieve traversal times that exceed what has been observed from long and aggregated short simulations without optimization. However, the approach also displayed inefficiencies with larger configurations when applying the NSGA-II algorithm out-of-the-box, as it does not scale well with increasing objectives and population sizes. It is concluded that the optimization approach yields promising results and offers a lot of pathways for future research and improvements.

By applying these improvements, which are mostly simple in nature, impressive resource savings or speedups up to a factor of multiple hundreds were observed in some test cases. These gains in efficiency can be leveraged and translated into novel ways to approach design automation and further improvements in validation techniques, such as the development of validation approaches based on machine learning. The results are promising and open interesting avenues for future research, which could revolutionize certain aspects of real-time network evaluation and design. Possible applications include tighter approximations of WCTT via simulation, more responsive and automated design systems, and better evaluation of the pessimism of analytical methods.

6.2 Directions for Future Work and Research

This section summarizes suggestions made to develop the proposed approaches further and provides an outlook on exciting future research paths.

6.2.1 Targeted Exploration of the Simulation State Space using NSO

One hypothesis constructed during this work but not directly tested or proven is that we can ”time travel” in (deterministic) simulation by using node start offsets to jump to specific simulation states of a long simulation using adjusted starting conditions. As discussed before, the modification of free variables allows for impacting the visited simulation states, particularly when all non-deterministic effects in the simulations are controlled or predictable.

One experiment that is not reported in this dissertation was conducted where the packet trace generated by the unfavorable scenario analysis as presented in Figure 4.2 could be reproduced in as short simulation of five milliseconds simulation time by determining corresponding node start offsets and applying minimal frame offsets to control the flow sending orders. This packet trace was generated by the unfavorable scenario analysis, which follows the transmission rules induced by QoS mechanism and network model at large. Consequently, the presented trace could just as well be a packet trace taken from any point in time from a simulation. By deriving offsets (and packet scheduling order), a short simulation can be generated that allows the reproduction of this part of the simulation trace. Assuming the ergodicity property, as discussed in Chapter 2, would render this equivalent to jumping to the moment at which this interference scenario is observed in a (potentially infinitely) long simulation trace. Jumping to certain points in the simulation state space is the essential idea behind exploring different node start offsets and flow scheduling orders, rather than slowly traversing them via long simulations with randomized clock drifts as illustrated in Figure 3.1.

At this time, it is not yet well understood in which areas the simulation states that yield actual worst-case traversal times are located within the simulation state space or how they can explicitly be reached. This could constitute an interesting topic of future research.

6.2.2 Increased Accessibility via Open-Source Simulation Software

As the simulation software used in this work is not openly available, the findings cannot be verified by the research community without added efforts of implementing the approach based on a different simulation software. Thus, one important future task will be to implement and evaluate the approach using open-source simulation software, such as OMNet++, which is currently the prominent choice in real-time Ethernet network research. This would allow us to evaluate the approach in an independent environment, and it can be made accessible to the entirety of the research community, providing a foundation for further research on validation approaches based on aggregation of short simulations.

6.2.3 Researching different Sampling Distributions for NSO

In this work, overlapping stratified ranges were proposed for exploring the node start offset parameter space via uniform sampling and as a biased initialization of the population of optimization algorithms. This strategy, in essence, simply represents an alternative distribution to uniform sampling over a fixed range. The stratified sampling approach provided an intuitive reasoning behind the observed effectiveness and proved beneficial over sampling from a uniform distribution.

There may, however, be other distributions that might yield better results. Consequentially, another direction for future research could be to explore other distributions, potentially in the context of different network configuration characteristics and properties. There may be different characteristics where one distribution may be beneficial over others, allowing the creation of adaptive or ensemble methods that improve efficiency by selecting the most appropriate distribution, depending on the specific characteristic of the system under investigation.

6.2.4 Approach Evaluation on additional Network Configurations

While the test cases that were used to evaluate the proposed evaluation paradigm were rather diverse in topology and traffic characteristics, they only covered a small portion of possible configurations in the domain. One desirable future work would be to evaluate the approach for other topologies, traffic characteristics, and network complexities. In that context, the effects of different specific characteristics could be investigated in isolation, for instance, the average number of hops, packet size, and flow periods. This opens a vast unexplored research area for develop optimized variations of the aggregation approach.

6.2.5 Analysis and Optimization of the different Speedup Variants

In Section 2.4.4, the different speedup factors involved in the efficiency of the aggregation approach have been discussed, and the increased parallelization potential was investigated in Section 3.4.3. The parallelization potential of the approach was not formally researched in this dissertation which could serve as an important future work. To that end, it may be beneficial to investigate the different factors contributing to the reduction of the parallelization speedup, as listed in Section 3.4.3.

6.2.6 Improved Optimization Algorithm for increased Scalability

As discussed in Chapter 5, the choice of NSGA-II as an optimization algorithm does not scale well to complex network architectures. However, NSGA-II is known to not scale well to high objective counts and many alternatives and improvements have been proposed to address this issue since the introduction of NSGA-II. One prospective future work would thus consist of evaluating state-of-the-art MOO algorithms that scale well to higher objective counts for their suitability for integration with the aggregation approach.

6.2.7 Applying Learning Methods for Improved Effectiveness

Machine Learning (ML) has proven beneficial in various applications across all domains. Techniques that have proven particularly effective for optimization problems, such as parameter tuning of simulation models, include Reinforcement Learning and Transfer Learning.

Neural Networks, and Deep Reinforcement Learning in particular, exhibit the ability to learn underlying problem structures inconceivable to humans. They allow the optimization of parameter selection to maximize certain metrics. In the context of this work, they could be used to replace the static distribution applied to sample the starting conditions by an adaptive ML-based sampler that learns to make better choices depending on the properties of the network configuration. Also, the optimization algorithm that was applied could potentially be replaced by an ML-based algorithm such as Reinforcement Learning, for instance.

Another ML technique worth mentioning is Graph Neural Networks, a specific type of

Neural Network that has enhanced capabilities of solving problems that exhibit a graph structure. Network communication is essentially such a graph-based problem, and ML was indeed successfully applied in the real-time networking domain before. For instance, Mai and Navet applied Graph Neural Networks to successfully predict the feasibility of TSN configurations in [69]. These techniques may also be suitable for the problems addressed in this dissertation and could yield potential future work for further enhancing the effectiveness of the aggregation approach.

Bibliography

[1] Core4INET. `https://omnetpp.org/download-items/Core4INET.html`. Accessed: 8 February 2024.

[2] IEEE Time-Sensitive Networking (TSN) Task Group. `https://1.ieee802.org/tsn/`. Accessed: 8 February 2024.

[3] INET framework release 4.4.0. `https://inet.omnetpp.org/2022-05-16-INET-4.4.0-released.html`. Accessed: 8 February 2024.

[4] OPNet website. `https://opnetprojects.com/opnet-network-simulator`. Accessed: 8 February 2024.

[5] Riverbed Modeler. `https://cms-api.riverbed.com/portal/community_home`. Accessed: 8 February 2024.

[6] RTaW-Pegase Homepage. `https://inet.omnetpp.org/`. Accessed: 8 February 2024.

[7] Time-Triggered Ethernet – A Powerful Network Solution for Multiple Purpose. `https://www.tttech.com/sites/default/files/documents/TTEthernet_Technical-Whitepaper_TTTech.pdf`. Accessed: 8 February 2024.

[8] ARINC 664 part 7, Avionics Full-Duplex Switched Ethernet(AFDX). `https://www.sae.org/standards/content/arinc664p7-1/`, 2009. Accessed: 8 February 2024.

[9] AS6802™. `https://www.sae.org/publications/technical-papers/content/AS6802/`, 2016. Accessed: 8 February 2024.

[10] Muhammad Adnan. *Exact worst-case communication delay analysis of AFDX network*. PhD thesis, 2013.

[11] Muhammad Adnan, Jean-Luc Scharbarg, Jérôme Ermont, and Christian Fraboul. Model for worst case delay analysis of an AFDX network using timed automata. In *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)*, pages 1–4. IEEE, 2010.

[12] Muhammad Adnan, Jean-Luc Scharbarg, Jérôme Ermont, and Christian Fraboul. An improved timed automata approach for computing exact worst-case delays of AFDX sporadic flows. In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, pages 1–8. IEEE, 2012.

[13] Muhammad Adnan, Jean-Luc Scharbarg, and Christian Fraboul. Minimizing the search space for computing exact worst-case delays of AFDX periodic flows. In *2011 6th IEEE International Symposium on Industrial and Embedded Systems*, pages 294–301. IEEE, 2011.

[14] M. Anand, S. Vestal, S. Dajani-Brown, and Insup Lee. Formal modeling and analysis of the AFDX frame management design. In *Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'06)*, pages 7 pp.–, 2006.

[15] Mohammad Ashjaei, Lucia Lo Bello, Masoud Daneshtalab, Gaetano Patti, Sergio Saponara, and Saad Mubeen. Time-Sensitive Networking in automotive embedded systems: State of the art and research opportunities. *Journal of systems architecture*, 117:102137, 2021.

[16] Susmita Bandyopadhyay and Ranjan Bhattacharya. *Discrete and continuous simulation: theory and practice*. CRC Press, 2014.

[17] Ananda Basu, Saddek Bensalem, Marius Bozga, Benoît Delahaye, Axel Legay, and Emmanuel Sifakis. Verification of an AFDX infrastructure using simulations and probabilities. In *International Conference on Runtime Verification*, pages 330–344. Springer, 2010.

[18] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach. *IEEE Transactions on Industrial informatics*, 6(4):521–533, 2010.

[19] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. Applying Trajectory approach with static priority queuing for improving the use of available AFDX resources. *Real-Time Systems*, 48(1):101–133, December 2011.

[20] Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, John Håkansson, Paul Pettersson, Wang Yi, and Martijn Hendriks. Uppaal 4.0. 2006.

[21] Nassima Benammar, Henri Bauer, Frédéric Ridouard, and Pascal Richard. Timing analysis of AVB Ethernet network using the forward end-to-end delay analysis. In *Proceedings of the 26th International Conference on Real-Time Networks and Systems*, pages 223–233, 2018.

[22] Nassima Benammar, Frédéric Ridouard, Henri Bauer, and Pascal Richard. Forward end-to-end delay analysis extension for FP/FIFO policy in AFDX networks. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2017.

[23] Nassima Benammar, Frédéric Ridouard, Henri Bauer, and Pascal Richard. Forward end-to-end delay for AFDX networks. *IEEE Transactions on Industrial Informatics*, 14(3):858–865, 2017.

[24] Marc Boyer, Nicolas Navet, and Marc Fumey. Experimental assessment of timing verification techniques for AFDX. In *ERTS 2012 - 6th European Congress on Embedded Real Time Software and Systems*, Toulouse, France, February 2012.

[25] Marc Boyer, Nicolas Navet, Xavier Olive, and Eric Thierry. The PEGASE project: precise and scalable temporal analysis for aerospace communication systems with network calculus. In *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, pages 122–136. Springer, 2010.

[26] Lelio Campanile, Marco Gribaudo, Mauro Iacono, Fiammetta Marulli, and Michele Mastroianni. Computer network simulation with ns-3: A systematic literature review. *Electronics*, 9(2):272, 2020.

[27] Gustavo Carneiro. NS-3: Network simulator 3. In *UTM Lab Meeting April*, volume 20, pages 4–5, 2010.

[28] Cheng-Shang Chang. *Performance guarantees in communication networks*. Springer Science & Business Media, 2000.

[29] Keyang Chang, Yimin Du, Min Liu, Jinglin Shi, Yiqing Zhou, and Yongkang Li. OS Packet Processing Mechanism Simulation Architecture for Enabling Digital Twins of Networks in ns-3. In *2023 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 186–193. IEEE, 2023.

[30] Xinjie Chang. Network simulations with opnet. In *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1*, pages 307–314, 1999.

[31] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul. Methods for bounding end-to-end delays on an AFDX network. In *18th Euromicro Conference on Real-Time Systems (ECRTS'06)*, pages 10 pp.–202, 2006.

[32] Edmund M Clarke, Thomas A Henzinger, Helmut Veith, Roderick Bloem, et al. *Handbook of model checking*, volume 10. Springer, 2018.

[33] Rene L Cruz. A calculus for network delay. Part I. Network elements in isolation. *IEEE Transactions on information theory*, 37(1):114–131, 1991.

[34] Rene L Cruz. A calculus for network delay. Part II. Network analysis. *IEEE Transactions on information theory*, 37(1):132–141, 1991.

[35] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[36] Jonas Diemer, Philip Axer, and Rolf Ernst. Compositional performance analysis in python with pycpa. *Proc. WATERS*, pages 27–32, 2012.

[37] Jonas Diemer, Jonas Rox, and Rolf Ernst. Modeling of Ethernet AVB networks for worst-case timing analysis. *IFAC Proceedings Volumes*, 45(2):848–853, 2012.

[38] Jonas Diemer, Daniel Thiele, and Rolf Ernst. Formal worst-case timing analysis of Ethernet topologies with strict-priority and AVB switching. In *7th IEEE International Symposium on Industrial Embedded Systems (SIES'12)*, pages 1–10. IEEE, 2012.

[39] Bradley Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[40] Jonathan Falk, David Hellmanns, Ben Carabelli, Naresh Nayak, Frank Dürr, Stephan Kehrer, and Kurt Rothermel. NeSTiNg: Simulating IEEE time-sensitive networking (TSN) in OMNeT++. In *2019 International Conference on Networked Systems (NetSys)*, pages 1–8. IEEE, 2019.

[41] A. Finzi, A. Mifdaoui, F. Frances, and E. Lochin. Incorporating TSN/BLS in AFDX for mixed-criticality applications: Model and timing analysis. In *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, pages 1–10, 2018.

[42] Christian M Fuchs, Advisors Stefan Schneele, and Er Klein. The evolution of avionics networks from ARINC 429 to AFDX. *Innovative Internet Technologies and Mobile Communications (IITM), and Aerospace Networks (AN)*, 65, 2012.

[43] Laurent George, Nicolas Rivierre, and Marco Spuri. *Preemptive and non-preemptive real-time uniprocessor scheduling.* PhD thesis, Inria, 1996.

[44] Wang Guo, Yanhong Huang, Jianqi Shi, Zhe Hou, and Yang Yang. A formal method for evaluating the performance of tsn traffic shapers using uppaal. In *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, pages 241–248. IEEE, 2021.

[45] Mukta Gupta, Ramakrishnan Durairajan, Meenakshi Syamkumar, Paul Barford, and Joel Sommers. pfs: Parallelized, flow-based network simulation. In *2015 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, pages 1–8. IEEE, 2015.

[46] J. Javier Gutiérrez, J. Carlos Palencia, and Michael González Harbour. Response time analysis in AFDX networks.

[47] Peter Heise, Fabien Geyer, and Roman Obermaisser. TSimNet: An industrial time sensitive networking simulation framework based on OMNeT++. In *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE, 2016.

[48] Mark D Hill and Michael R Marty. Amdahl's law in the multicore era. *Computer*, 41(7):33–38, 2008.

[49] Gerard J. Holzmann. The model checker SPIN. *IEEE Transactions on software engineering*, 23(5):279–295, 1997.

[50] Max Hort and Federica Sarro. The effect of offspring population size on NSGA-II: a preliminary study. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '21. ACM, July 2021.

[51] Oana Hotescu, Katia Jaffrès-Runser, Jean-Luc Scharbarg, and Christian Fraboul. Multiplexing Avionics and additional flows on a QoS-aware AFDX network. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 282–289. IEEE, 2019.

[52] Jahanzaib Imtiaz, Jürgen Jasperneite, and Lixue Han. A performance study of Ethernet Audio Video Bridging (AVB) for Industrial real-time communication. In *2009 IEEE Conference on Emerging Technologies & Factory Automation*, pages 1–8. IEEE, 2009.

[53] Junhui Jiang, Yuting Li, Seung Ho Hong, Aidong Xu, and Kai Wang. A time-sensitive networking (TSN) simulation model based on OMNET++. In *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 643–648. IEEE, 2018.

[54] Patrick Keller and Nicolas Navet. Approximation of worst-case latencies in real-time Ethernet networks via aggregated short simulations: how to select the relevant parameters. Submitted to DATE2024; rejected.

[55] Patrick Keller and Nicolas Navet. Approximation of Worst-Case Traversal Times in Real-Time Ethernet Networks: Exploring the Potential of Many-Objective Optimization for Simulation Aggregation. Submitted to WFCS2024, accepted.

[56] Patrick Keller and Nicolas Navet. Approximating WCRT through the Aggregation of Short Simulations with Different Initial Conditions: Application to TSN. In *Proceedings of the 30th International Conference on Real-Time Networks and Systems*, RTNS '22, page 196–206, New York, NY, USA, 2022. Association for Computing Machinery.

[57] Georges Kemayo, Frédéric Ridouard, Henri Bauer, and Pascal Richard. Optimism due to serialization in the trajectory approach for switched ethernet networks. In

Proc. of Int. Conf. on Junior Researcher Workshop on Real-Time Computing (JR-WRTC), pages 13–16, 2013.

[58] Georges Kemayo, Frédéric Ridouard, Henri Bauer, and Pascal Richard. Optimistic problems in the trajectory approach in fifo context. In *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, pages 1–8. IEEE, 2013.

[59] Georges Kemayo, Frédéric Ridouard, Henri Bauer, and Pascal Richard. A forward end-to-end delays analysis for packet switched networks. In *Proceedings of the 22nd International Conference on Real-Time Networks and Systems*, pages 65–74, 2014.

[60] Hermann Koptez. Real-time systems: design principles for distributed embedded applications. *Kluwer Academic Publisher*, 1997.

[61] Dennis Krummacker and Luca Wendling. TSN Simulation: Time-Aware Shaper implemented in ns-3. In *Proceedings of the 2020 Workshop on Next Generation Networks and Applications (NGNA 2020), Kaiserslautern, Germany*, pages 16–17, 2020.

[62] Robert A. LaBudde and Michael R. Chernick. An introduction to bootstrap methods with applications to R, 2011.

[63] Leslie Lamport. Real-time model checking is really simple. In *Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, pages 162–175. Springer, 2005.

[64] Michaël Lauer. *Une méthode globale pour la vérification d'exigences temps réel: application à l'Avionique Modulaire Intégrée*. PhD thesis, 2012.

[65] Jean-Yves Le Boudec. Application of network calculus to guaranteed service networks. *IEEE Transactions on Information theory*, 44(3):1087–1096, 1998.

[66] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, volume 2050. 06 2004.

[67] Xiaoting Li, Olivier Cros, and Laurent George. The Trajectory approach for AFDX FIFO networks revisited and corrected. In *2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 1–10. IEEE, 2014.

[68] Xiaoting Li, Jean-Luc Scharbarg, and Christian Fraboul. Analysis of the pessimism of the Trajectory approach for upper bounding end-to-end delay of sporadic flows sharing a switched Ethernet network. In *RTNS*, pages 149–158. Citeseer, 2011.

[69] Tieu Long Mai and Nicolas Navet. Improvements to deep-learning-based feasibility prediction of switched ethernet network configurations. In *29th International Conference on Real-Time Networks and Systems*, pages 89–99, 2021.

[70] Jin Lv, Yongxin Zhao, Xi Wu, Yongjian Li, and Qiang Wang. Formal analysis of TSN scheduler for real-time communications. *IEEE Transactions on Reliability*, 70(3):1286–1294, 2020.

[71] Etienne Mabille, Marc Boyer, Loïc Fejoz, and Stephan Merz. Certifying Network Calculus in a Proof Assistant. In *EUCASS-5th European Conference for Aeronautics and Space Sciences*, 2013.

[72] Marcus Märtens and Dario Izzo. The asynchronous island model and NSGA-II: study of a new migration operator and its performance. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, GECCO '13. ACM, July 2013.

[73] Steven Martin and Pascale Minet. Holistic and trajectory approaches for distributed non-preemptive FP/DP* scheduling. In *International Conference on Networking*, pages 296–305. Springer, 2005.

[74] Steven Martin and Pascale Minet. Schedulability analysis of flows scheduled with FIFO: application to the expedited forwarding class. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, pages 8–pp. IEEE, 2006.

[75] Steven Martin and Pascale Minet. Worst case end-to-end response times of flows scheduled with FP/FIFO. In *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)*, pages 54–54. IEEE, 2006.

[76] Steven Martin, Pascale Minet, and Laurent George. The trajectory approach for the end-to-end response times with non-preemptive fp/edf. In *Software Engineering Research and Applications: Second International Conference, SERA 2004, Los Angeles, CA, USA, MAY 5-7, 2004, Revised Selected Papers 2*, pages 229–247. Springer, 2006.

[77] Sara Medlej. *Scalable Trajectory Approach for ensuring deterministic guarantees in large networks*. PhD thesis, Université Paris Sud-Paris XI, 2013.

[78] Sara Medlej, Steven Martin, and Jean-Marie Cottin. Identifying sources of pessimism in the trajectory approach with FIFO scheduling. In *Embedded Real Time Software and Systems (ERTS2012)*, 2012.

[79] Jihyeon Min and Youngil Park. Application of 10BASE-T1S Ethernet in Delay-Sensitive Vehicular Networks. In *2023 IEEE Vehicular Networking Conference (VNC)*, pages 57–60. IEEE, 2023.

[80] Nicolas Navet, Jan Seyler, and Jörn Migge. Timing verification of real-time automotive Ethernet networks: what can we expect from simulation? In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, TOULOUSE, France, January 2016.

[81] Eugene David Ngangue Ndih and Soumaya Cherkaoui. Simulation methods, techniques and tools of computer systems and networks. In *Modeling and simulation of computer networks and systems*, pages 485–504. Elsevier, 2015.

[82] Maryam Pahlevan and Roman Obermaisser. Evaluation of time-triggered traffic in time-sensitive networks using the opnet simulation framework. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pages 283–287. IEEE, 2018.

[83] Kalyan Perumalla, Maximilian Bremer, Kevin Brown, Cy Chan, Stephan Eidenbenz, K Scott Hemmert, Adolfy Hoisie, Benjamin Newton, James Nutaro, Tomas Oppelstrup, et al. Computer science research needs for parallel discrete event simulation (PDES). Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2022.

[84] Dimitrios Piromalis and Antreas Kantaros. Digital twins in the automotive industry: The road toward physical-digital convergence. *Applied System Innovation*, 5(4):65, 2022.

[85] Maurice H Quenouille. Approximate tests of correlation in time-series 3. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 45, pages 483–484. Cambridge University Press, 1949.

[86] George F Riley and Thomas R Henderson. The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer, 2010.

[87] G.F. Riley, R.M. Fujimoto, and M.H. Ammar. A generic framework for parallelization of network simulations. In *MASCOTS '99. Proceedings of the Seventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 128–135, 1999.

[88] Soheil Samii, Sergiu Rafiliu, Petru Eles, and Zebo Peng. A Simulation Methodology for Worst-Case Response Time Estimation of Distributed Real-Time Systems. In *2008 Design, Automation and Test in Europe*, pages 556–561, 2008.

[89] Luca Santinelli and Liliana Cucu-Grosjean. Toward probabilistic real-time calculus. *ACM SIGBED Review*, 8(1):54–61, 2011.

[90] Luca Santinelli and Liliana Cucu-Grosjean. A probabilistic calculus for probabilistic real-time systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 14(3):1–30, 2015.

[91] Youhwan Seol, Doyeon Hyeon, Junhong Min, Moonbeom Kim, and Jeongyeup Paek. Timely survey of time-sensitive networking: Past and future directions. *Ieee Access*, 9:142506–142527, 2021.

[92] Joseph Sifakis and David Harel. Trustworthy Autonomous System Development. *ACM Transactions on Embedded Computing Systems*, 22(3):1–24, April 2023.

[93] N. Srinivas and Kalyanmoy Deb. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, 09 1994.

[94] Xueqian Tang, Qiao Li, Guangshan Lu, and Huagang Xiong. A Revised Trajectory Approach for the Worst-Case Delay Analysis of an AFDX Network. *Ieee Access*, 7:142564–142573, 2019.

[95] Daniel Thiele, Philip Axer, and Rolf Ernst. Improving formal timing analysis of switched ethernet by exploiting FIFO scheduling. In *Proceedings of the 52nd Annual Design Automation Conference*, pages 1–6, 2015.

[96] Daniel Thiele, Philip Axer, Rolf Ernst, and Jan Seyler. Improving formal timing analysis of switched ethernet by exploiting traffic stream correlations. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, pages 1–10, 2014.

[97] Lothar Thiele, Samarjit Chakraborty, and Martin Naedele. Real-time calculus for scheduling hard real-time systems. In *2000 IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 4, pages 101–104. IEEE, 2000.

[98] Ye Tian, Langchun Si, Xingyi Zhang, Ran Cheng, Cheng He, Kay Chen Tan, and Yaochu Jin. Evolutionary Large-Scale Multi-Objective Optimization: A Survey. *ACM Computing Surveys*, 54(8):1–34, October 2021.

[99] Ken Tindell. *Fixed Priority scheduling of hard real-time systems*. PhD thesis, University of York, 1994.

[100] Ken Tindell and John Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and microprogramming*, 40(2-3):117–134, 1994.

[101] Stavros Tripakis and Costas Courcoubetis. Extending Promela and Spin for real time. In *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, pages 329–348. Springer, 1996.

[102] John Tukey. Bias and confidence in not quite large samples. *Ann. Math. Statist.*, 29:614, 1958.

[103] John Wilder Tukey et al. *Exploratory data analysis*, volume 2. Springer, 1977.

[104] Andras Varga. OMNeT++. In *Modeling and tools for network simulation*, pages 35–59. Springer, 2010.

[105] Ernesto Wandeler, Lothar Thiele, Marcel Verhoef, and Paul Lieverse. System architecture evaluation using modular performance analysis: a case study. *International Journal on Software Tools for Technology Transfer*, 8:649–667, 2006.

[106] Jain J Wang and Marc Abrams. Determining initial states for time-parallel simulations. In *Proceedings of the seventh workshop on Parallel and distributed simulation*, pages 19–26, 1993.

[107] Klaus Wehrle, Mesut Günes, and James Gross. *Modeling and tools for network simulation*. Springer Science & Business Media, 2010.

[108] Dong Xie, Jiang Li, and Huisheng Gao. Comparison and Analysis of Simulation methods for TSN Performance. In *IOP Conference Series: Materials Science and Engineering*, volume 768, page 052061. IOP Publishing, 2020.

[109] Qingfei Xu and Xinyu Yang. Performance Evaluation on Packet Transmission for AFDX Networks Using Optimized Forward End-to-End Delay Analysis. *International Journal of Aeronautical and Space Sciences*, pages 1–15, 2020.

[110] Peng Zhang, Yu Liu, Jianqi Shi, Yanhong Huang, and Yongxin Zhao. A feasibility analysis framework of time-sensitive networking using real-time calculus. *IEEE Access*, 7:90069–90081, 2019.

[111] Raffaele Zippo and Giovanni Stea. Nancy: An efficient parallel Network Calculus library. *SoftwareX*, 19:101178, July 2022.

Appendix A
Statistical Tools

A.1 Bootstrapping

Bootstrapping refers to a family of statistical techniques that rely on resampling with replacement. They are used to estimate the distribution of a statistic or metric, particularly when the number of samples is limited. This is often the case when evaluation is costly, like in simulation for instance. The bootstrapping technique suggests to resample from the original dataset by sampling with replacements and uses an equiprobable sampling distribution. By creating many such resamplings, known as ”bootstrap sample”, the distribution of the statistical measure can be studied, reflecting the statistical distribution of the original dataset. The bootstrapping technique allows the computation of different statistical measures and metrics like mean, average, bias, variance, confidence intervals, and others.

In 1979, ”the bootstrap” was first published by Efron in [39] as a generalization of the ”jackknife”, based on principles introduced by Quenouille in 1951 in [85] and developed by Tukey in [102]. It is essentially a resampling method based on systematically leaving out observations to approximate the bias and variance of a dataset. The work by Efron allowed to alleviate certain limitations of the ”jackknife” method and improve flexibility, accuracy, and applicability. The improvements include utilization of a large variety of statistical measures, increased robustness and applicability to larger datasets. A practical introduction and overview of different bootstrapping variants is given by LaBudde and Chernik in [62].

In this dissertation, bootstrapping is used to estimate the general behavior of short simulations in comparison to long simulations in Chapter 3. This is done by resampling with replacement from the dataset of short simulations in order to create ”resampled experiments” and compare them to the set of long simulation experiments. For each ”resampled experiment”, a number of short simulations are sampled such that the total simulation time budget is equal to the simulation time of a long simulation. This is valid as all executions of the short simulations are independent of each other, and the starting conditions are sampled equiprobably. As a consequence, each of the resampled short simulations could ”naturally” occur when running a short simulation aggregation experiment.

This way, a large number of virtual aggregation experiments could be produced without investing extensive simulation efforts. However, one notable limitation of this approach is that it does not allow the discovery of new data, notably higher or lower end-to-end delays than are present in the original dataset from which it is resampled. This limitation is not significant in the experiments where resampling was used, as the focus is on the general performance trend of short simulation aggregation. More details about how the resampling was carried out can be found in [56].

A.2 Probability Distribution Function (PDF)

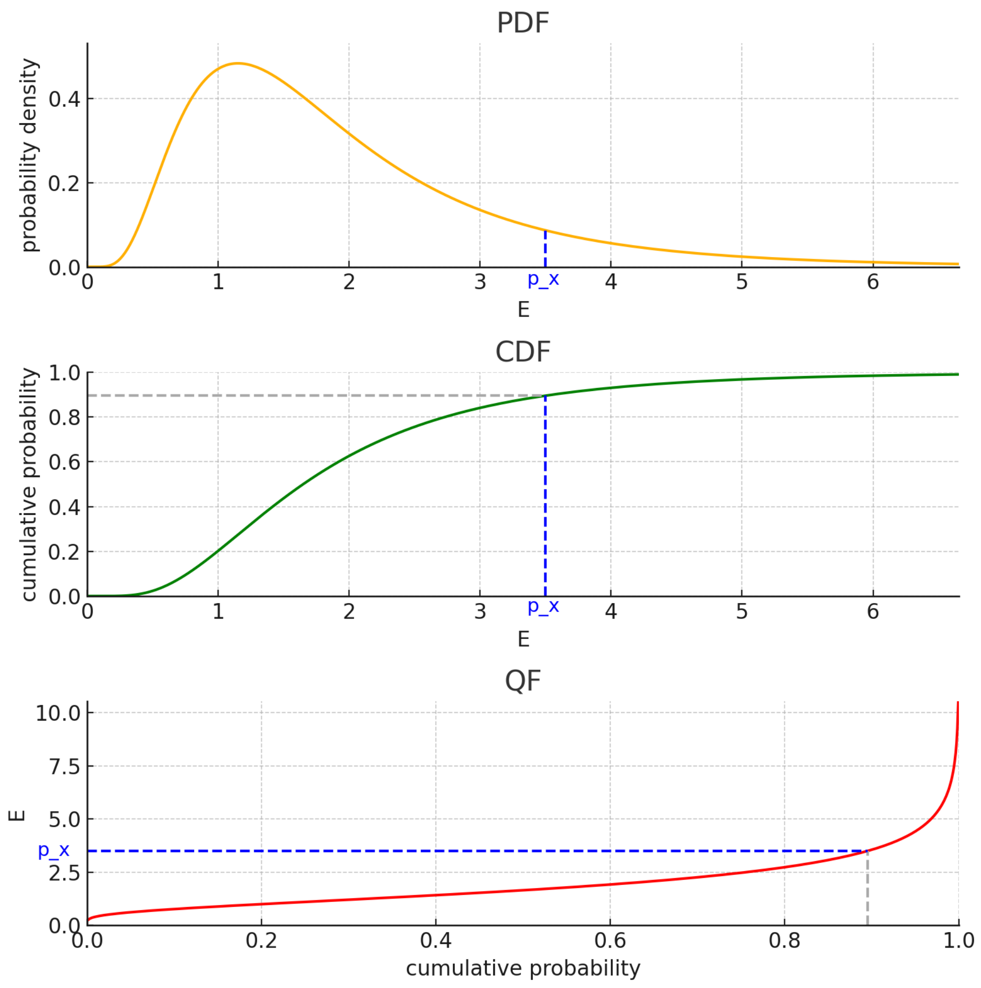


Figure A.1: **Example PDF, CDF and QF**
The figure shows an example of a probability distribution function and the corresponding cumulative probability function and quantile functions. Further, an example value $p_x := 3.5$ is annotated, which represents (approximately) the value of the 90% quantile in this example.

A probability distribution function (PDF) represents the probability distribution of a random variable with respect to its possible values. For instance, Figure A.1 shows a continuous PDF where the x-axis displays the value of a random variable E and the y-axis describes the probability of observing a specific value. As the probability of observing all values is one, the integral (or area under the curve) of the PDF is equal to one.

The cumulative distribution function (CDF) is related to the PDF and describes the cumulative probability of the variable E being less or equal to the value p_x. The CDF is obtained by integrating the PDF from negative infinity to p_x and represents the probability of the random variable E taking a value smaller than or equal to p_x.

The quantile function (QF) is the inverse of the CDF and represents the value of p_x such that the quantile portion of the dataset is below or equal to the value p_x. All three functions are illustrated in Figure A.1, based on an example log-normal function as the PDF.

A.3 Quantiles

Quantiles are statistical measures that divide a dataset into equally sized contiguous intervals. We can define a p quantile as a value x_p with $0 < p <= 1$, such that a portion p of the dataset lies below (or equals) x_p and a $(1-p)$ portion of the dataset lies above x_p.
As an example, given a dataset $\{0, 2, 4, 5, 6, 8, 10\}$, the first 50% quantile contains values $\{0, 2, 4, 5\}$ and the second 50% quantile contains the remaining values. Figure A.1 illustrates a 90% quantile of a continuous dataset represented by a log-normal function and comprises all the observations up to approximately the value of $p_x := 3.5$.

In worst-case delay evaluation, we are typically interested in very rare events. Thus, rather large quantiles are considered. For instance, a typical quantile of interest is the 99.9999% quantile ($1 - 0.1^6$ quantile), which represents events that occur once in one million observations. We defined these quantiles as exponentials as follows

$$QE := 1 - 0.1^E$$

where E is the exponent of the fraction. It should be noted there is no standard notation for quantiles and it can be encountered for other quantiles or quartiles, for instance Q3 is a popular notation to describe the third quartile or 75% quantile but it can also be encountered as the 99.9% quantile.

A.4 Boxplot

Boxplots, introduced by Tukey in 1977 in [103], are a representation of statistical data that display the distribution of the dataset based on its quartiles. A quartile is a quantile of a

multiple of 25% of the dataset as introduced in the section A.3.
The statistical measures displayed by a box plot include the minimum, the first quantile (or 25% quantile), the median (or second quartile, 50% quantile), the third quartile (or 75% quantile), and the maximum. The boxplot extends from the whiskers that extend up to 1.5 times the inter-quartile range (the range between the 25% quantile and the 75% quantile), and everything above or below these values is considered outliers.
As depicted in Figure A.2, the whiskers are displayed at horizontal lines at the top and bottom of the boxplot, the inter-quartile range is displayed as a box containing a horizontal line which represents the median and the outliers are typically represented as dots or small circles. Box plots allow the visualization and comparison of the central tendency of distributions of datasets, as is done in this dissertation in Chapter 3.

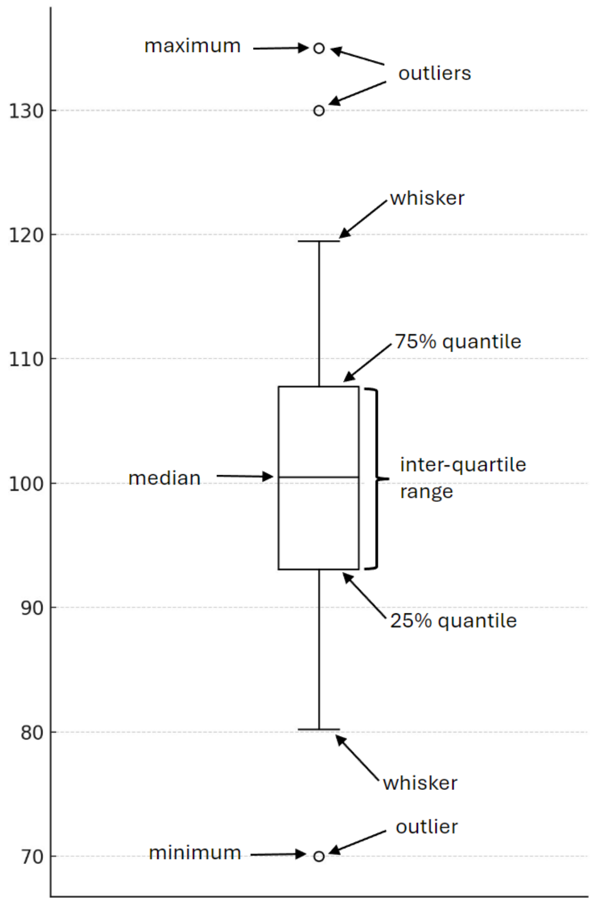


Figure A.2: **Example Boxplot**
An example boxplot that shows a data distribution in a range from 70 to 135 with 3 outliers. All relevant measures, like minimum, maximum, median, and quartiles, are annotated accordingly.

Appendix B
Test Case Details

This chapter provides additional information on each use case to improve the reproducibility of the experiments and provide deeper insights into the diverse configuration characteristics. Each of the following sections will provide plots showing the distribution of flow receptions, frame sizes, flow periods (or minimum distance between emissions for sporadic flows), and reception deadlines.

The following sections present the details of the test cases presented in Chapters 3.3.2, 4.3.1 and 5.3.1. Please note that the automotive FP/FIFO and FP/FIFO+CBS configurations presented in Chapter 3.3.2 are identical as they only differ by the used QoS mechanisms, and their traffic characteristics are shown in Section B.1. In Section B.2 the traffic characteristics of the FIFO version of the automotive configuration in Chapters 4.3.1 and 5.3.1, which substitute the CBS version used in Chapter 3.3.2, are presented. In the case of sporadic flows, the values reported as period refer to the minimum distance between packet emissions. The lowest priority flows sometimes represent best-effort traffic and may not have deadlines for that reason. In that case, they are reported as zero.

B.1 Automotive Configuration with FP/FIFO traffic

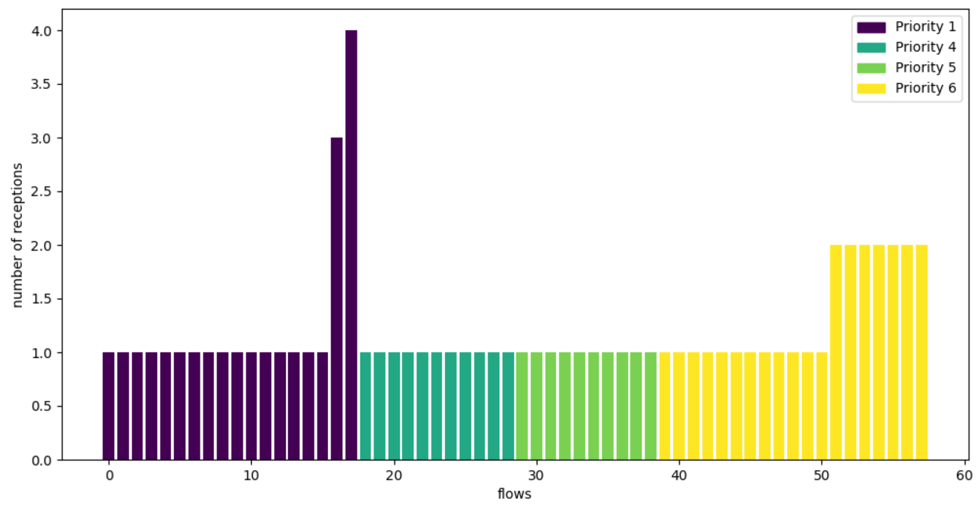


Figure B.1: **Automotive FP/FIFO - Flow receptions per flow.**
The plot shows the number of receptions (on the y-axis) per flow (on the x-axis). Flows are sorted and colored by ascending priority and ordered by ascending number of receptions.

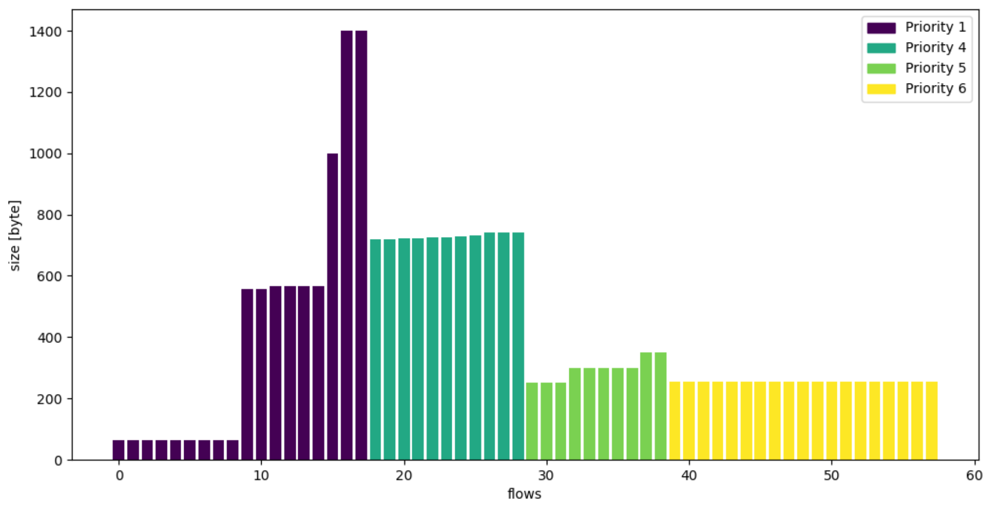


Figure B.2: **Automotive FP/FIFO - Frame sizes per flow.**
The plot shows the frame sizes in byte (on the y-axis) per flow (on the x-axis). Flows are sorted and colored by ascending priority and ordered by ascending frame sizes.

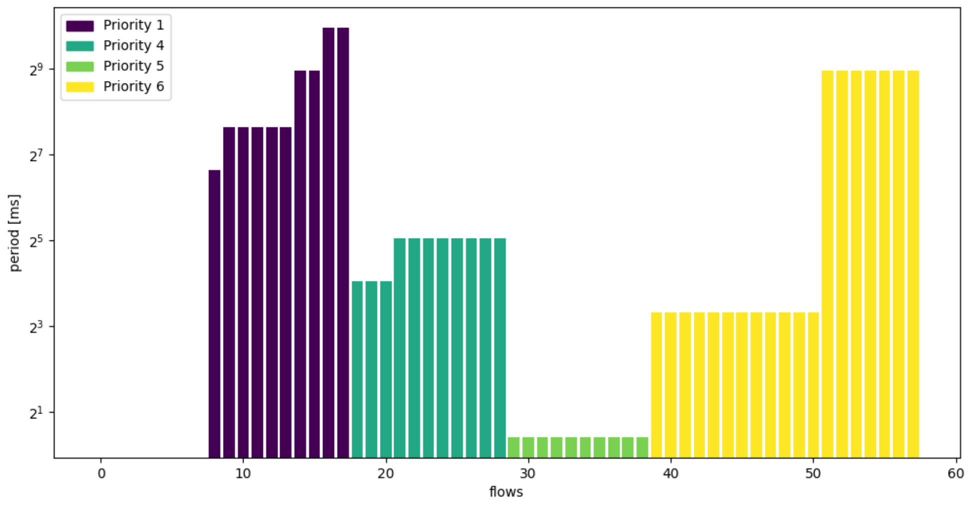


Figure B.3: **Automotive FP/FIFO - Frame periods per flow.**
The plot shows the frame periods in milliseconds (on the y-axis) per flow (on the x-axis). Flows are sorted and colored by ascending priority and ordered by ascending periods. The ACK and DATA components of TFTP flows have no periods as they are dialogue-based and are reported as zero.

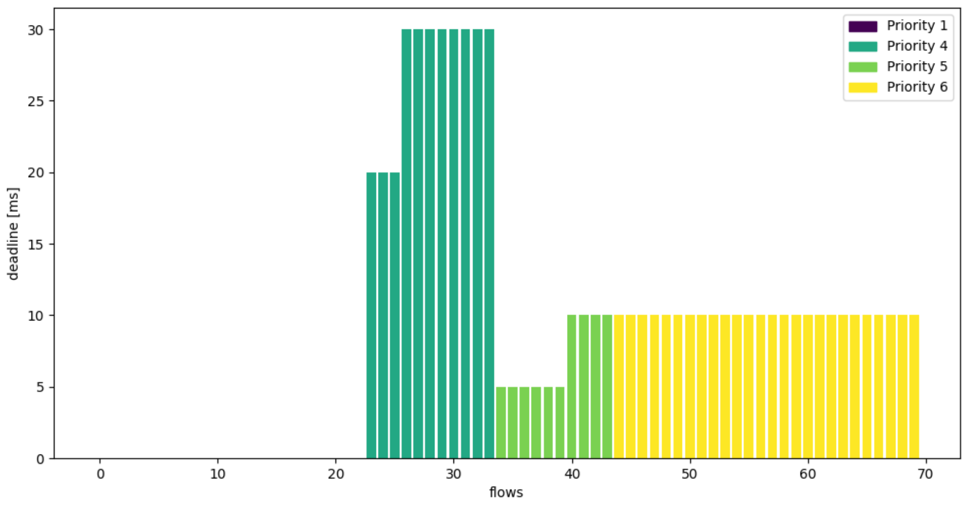


Figure B.4: **Automotive FP/FIFO - Deadlines per flow reception.**
The plot shows the deadline in milliseconds (on the y-axis) per flow reception (on the x-axis). Flow receptions are sorted and colored by ascending flow priority and ordered by ascending deadlines. The lowest priority flows are best effort, which have no deadline, and are reported as zeros in this plot.

B.2 Automotive Configuration with FIFO traffic

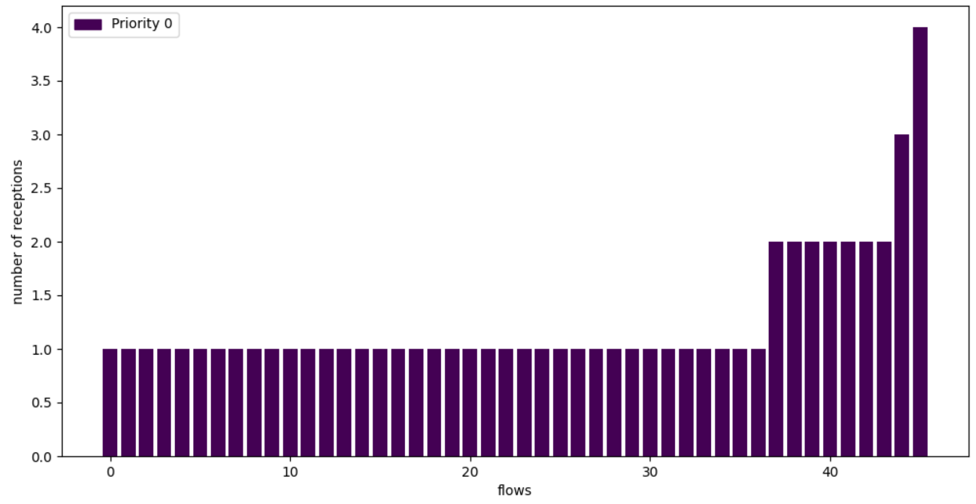


Figure B.5: **Automotive FIFO - Flow receptions per flow.**
The plot shows the number of receptions (on the y-axis) per flow (on the x-axis). Flows are ordered by ascending number of receptions.

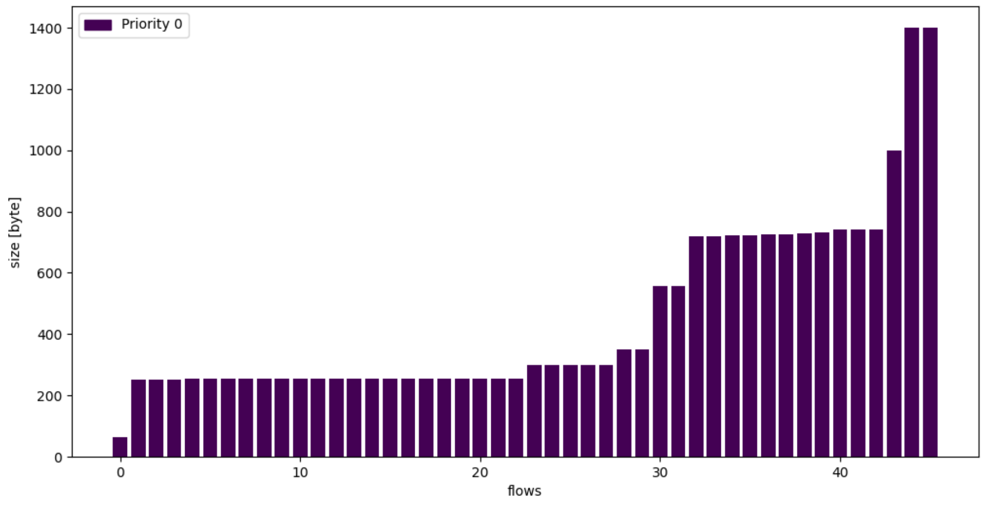


Figure B.6: **Automotive FIFO - Frame sizes per flow.**
The plot shows the frame sizes in bytes (on the y-axis) per flow (on the x-axis). Flows are ordered by ascending frame sizes.

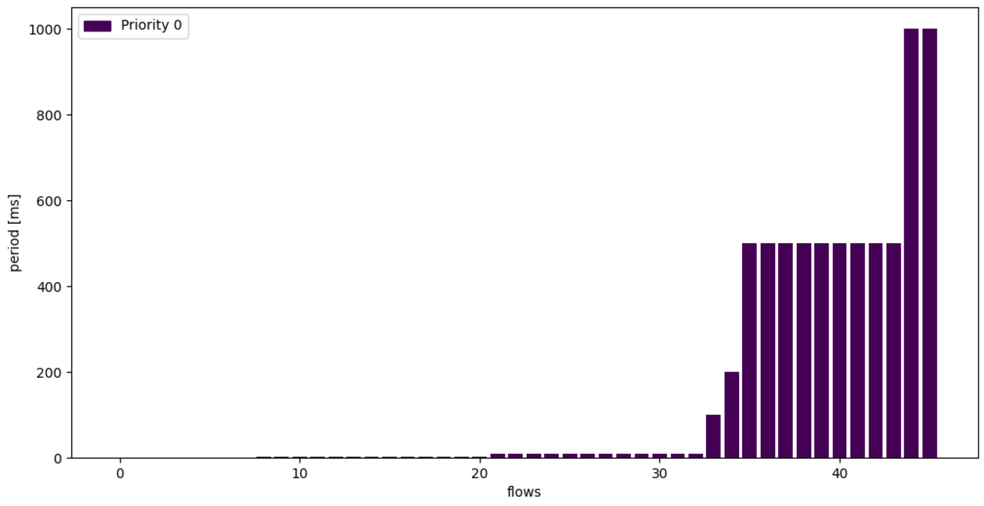


Figure B.7: **Automotive FIFO - Frame periods per flow.**
The plot shows the frame periods in milliseconds (on the y-axis) per flow (on the x-axis). Flows are ordered by ascending periods. The lower blocks represent flows with 0.33 ms, 0.9 to 1.33 ms and 10 ms periods.

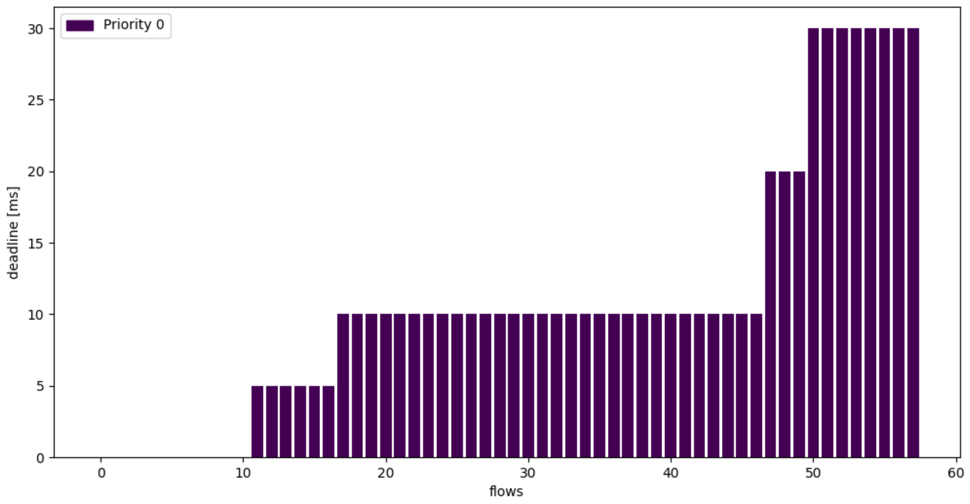


Figure B.8: **Automotive FIFO - Deadlines per flow reception.**
The plot shows the deadline in milliseconds (on the y-axis) per flow reception (on the x-axis). Flow receptions are ordered by ascending deadlines. Some best-effort flows, that have no deadlines, are included and are reported as zero.

B.3 Avionics Configuration with FP/FIFO traffic

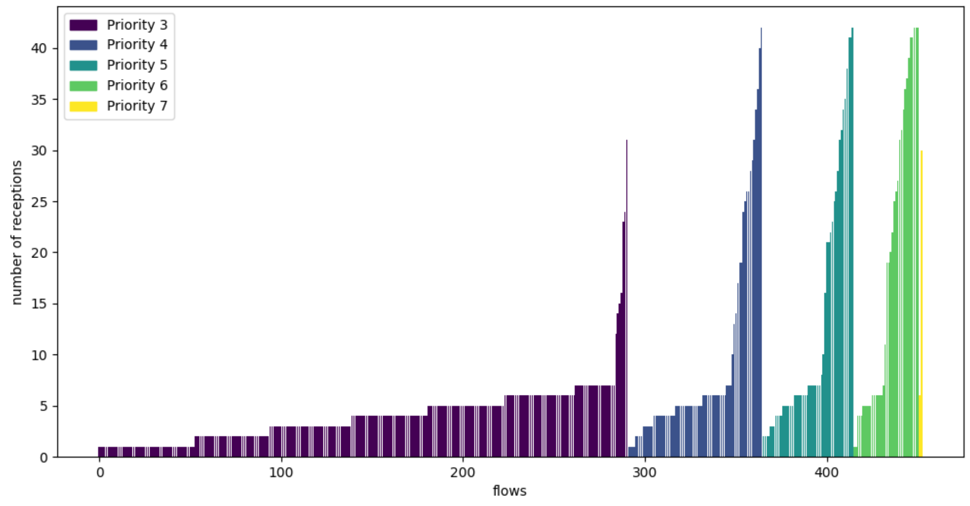


Figure B.9: **Avionics FP/FIFO - Flow receptions per flow.**
The plot shows the number of receptions (on the y-axis) per flow (on the x-axis). Flows are sorted and colored by ascending priority and ordered by ascending number of receptions.

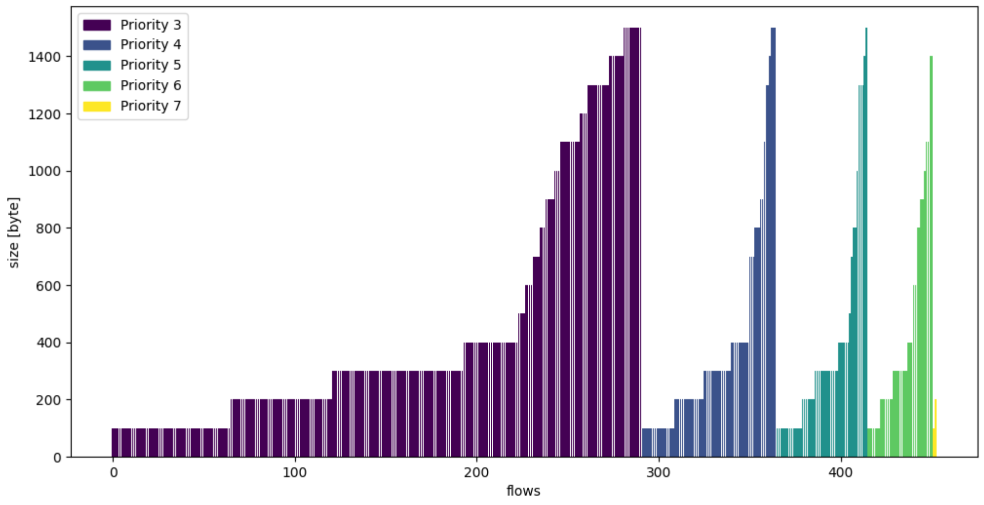


Figure B.10: **Avionics FP/FIFO - Frame sizes per flow.**
The plot shows the frame sizes in bytes (on the y-axis) per flow (on the x-axis). Flows are sorted and colored by ascending priority and ordered by ascending frame sizes.

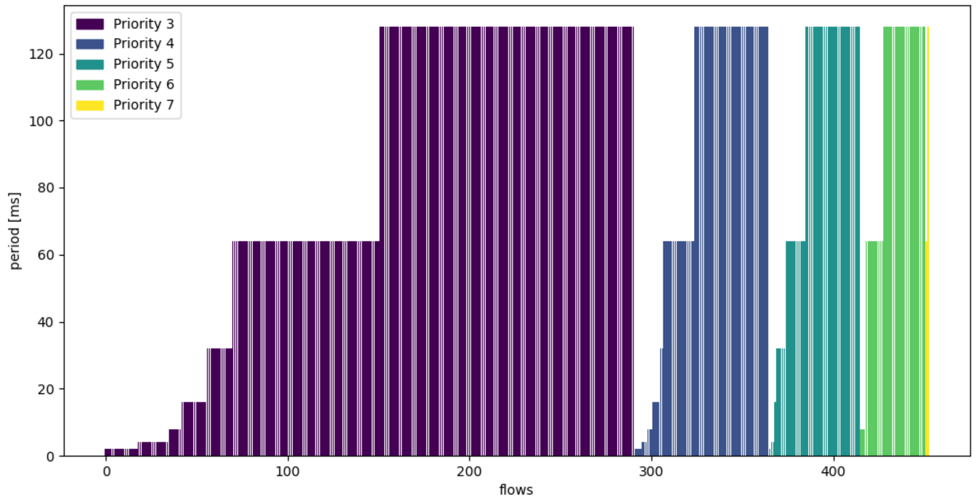


Figure B.11: **Avionics FP/FIFO - Frame periods per flow.**
The plot shows the frame periods in milliseconds (on the y-axis) per flow (on the x-axis). Flows are sorted and colored by ascending priority and ordered by ascending periods.

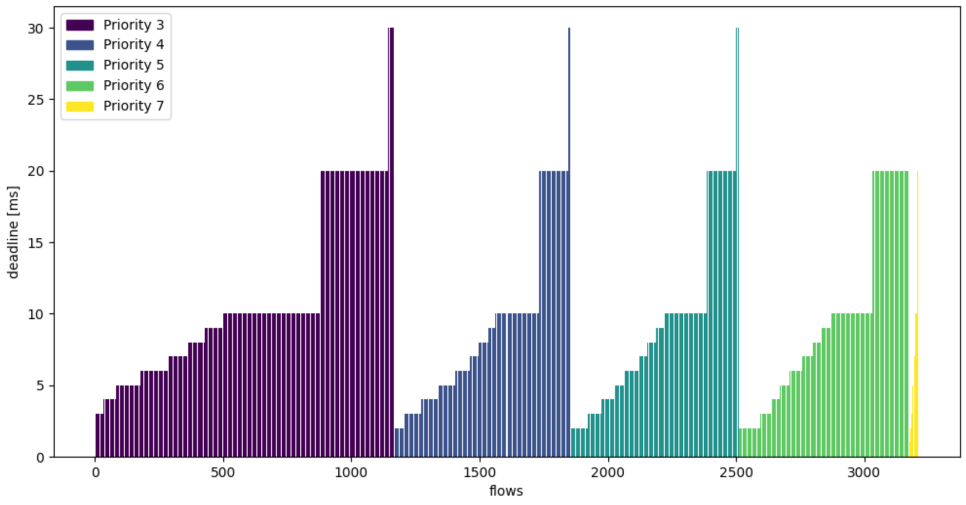


Figure B.12: **Avionics FP/FIFO - Deadlines per flow reception.**
The plot shows the deadline in milliseconds (on the y-axis) per flow reception (on the x-axis). Flow receptions are sorted and colored by ascending flow priority and ordered by ascending deadlines.

B.4 Avionics Configuration with FIFO traffic

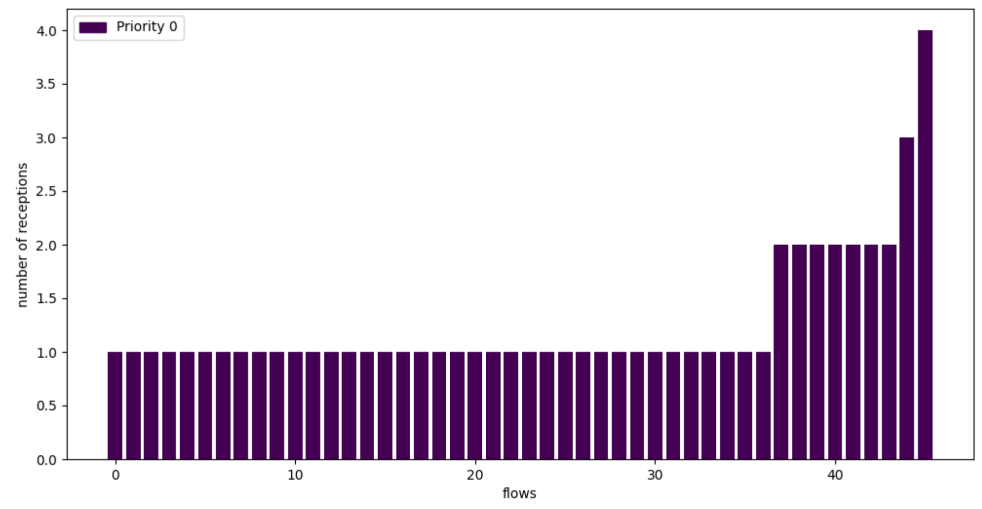


Figure B.13: **Avionics FIFO - Flow receptions per flow.**
The plot shows the number of receptions (on the y-axis) per flow (on the x-axis). Flows are ordered by ascending number of receptions.

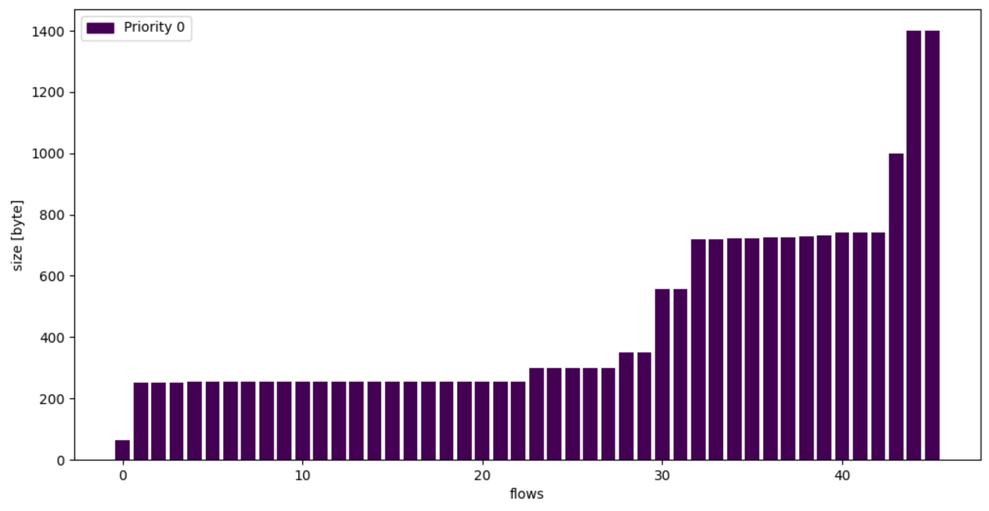


Figure B.14: **Avionics FIFO - Frame sizes per flow.**
The plot shows the frame sizes in bytes (on the y-axis) per flow (on the x-axis). Flows are ordered by ascending frame sizes.

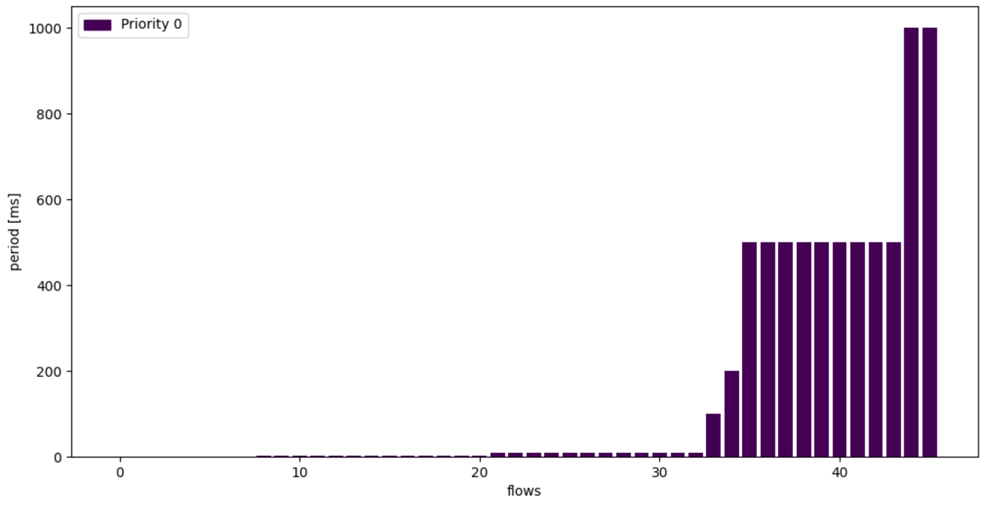


Figure B.15: **Avionics FIFO - Frame periods per flow.**
The plot shows the frame periods in milliseconds (on the y-axis) per flow (on the x-axis). Flows are ordered by ascending periods. The lower blocks represent flows with periods from 0.3 to 0.33ms, 0.9 to 1.66 ms and 10 ms.

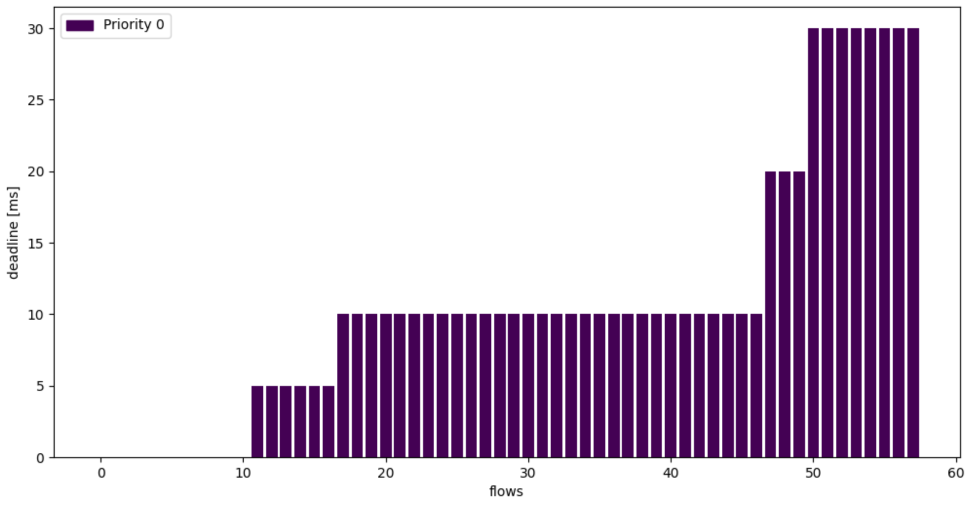


Figure B.16: **Avionics FIFO - Deadlines per flow reception.**
The plot shows the deadline in milliseconds (on the y-axis) per flow reception (on the x-axis). Flow receptions are ordered by ascending deadlines. Best-effort flows, which have no deadline, are shown as zero.

B.5 Space Launcher Configuration

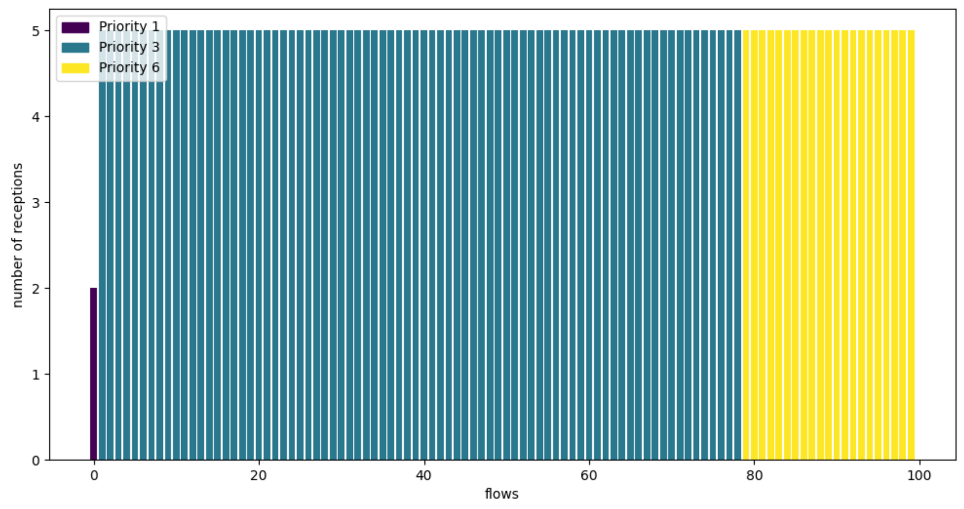


Figure B.17: **Space Launcher FP/FIFO - Flow receptions per flow.** The plot shows the number of receptions (on the y-axis) per flow (on the x-axis). Flows are sorted and colored by ascending priority and ordered by ascending number of receptions.

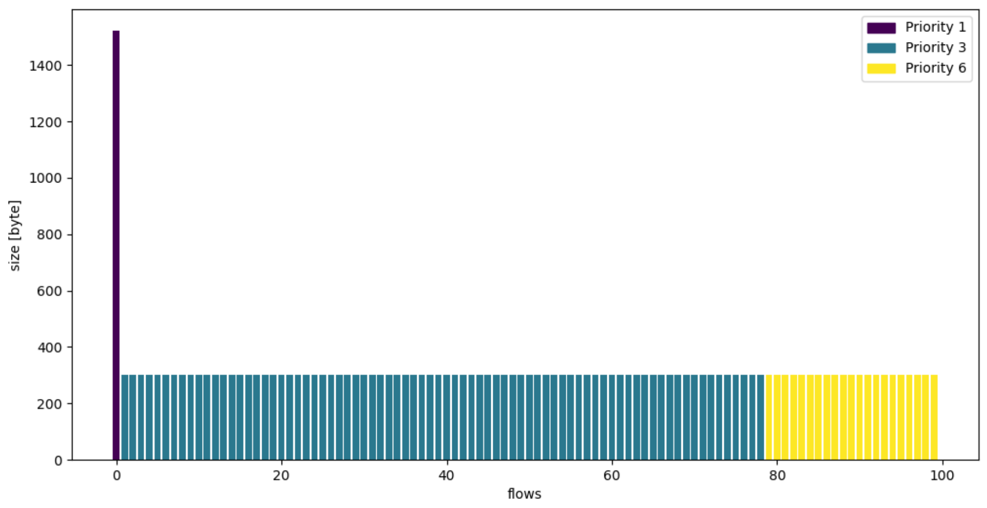


Figure B.18: **Space Launcher FP/FIFO - Frame sizes per flow.** The plot shows the frame sizes in bytes (on the y-axis) per flow (on the x-axis). Flows are sorted and colored by ascending priority and ordered by ascending frame sizes.

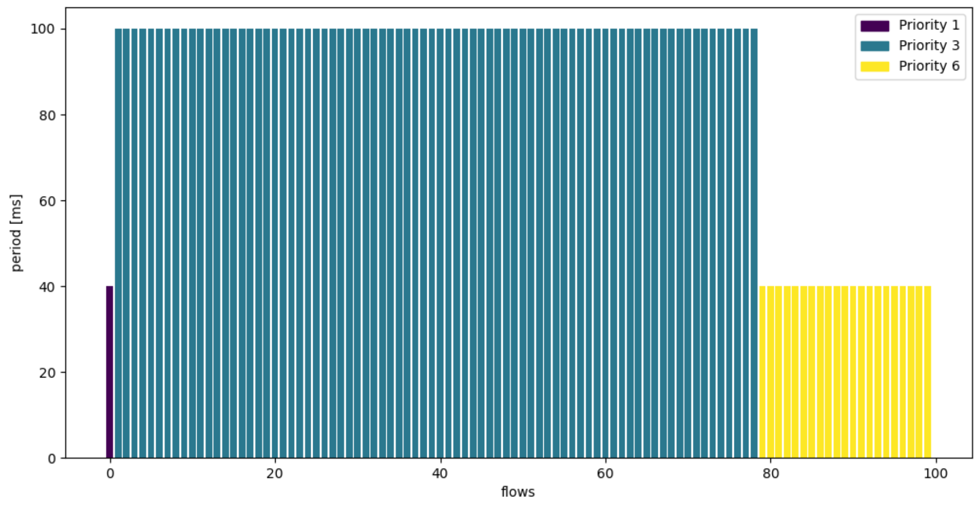


Figure B.19: **Space Launcher FP/FIFO - Frame periods per flow.**
The plot shows the frame periods in milliseconds (on the y-axis) per flow (on the x-axis). Flows are sorted and colored by ascending priority and ordered by ascending periods.

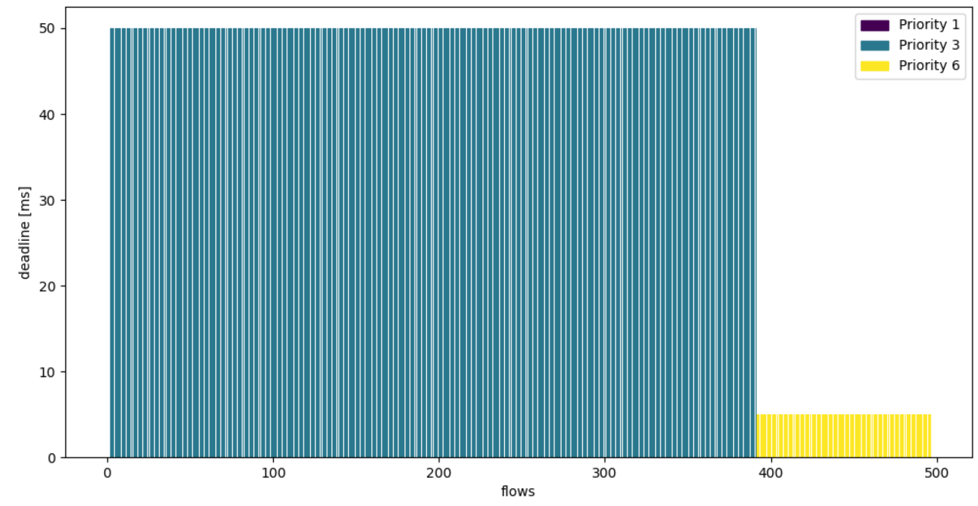


Figure B.20: **Space Launcher FP/FIFO - Deadlines per flow reception.**
The plot shows the deadline in milliseconds (on the y-axis) per flow reception (on the x-axis). Flow receptions are sorted and colored by ascending flow priority and ordered by ascending deadlines.