JOINT SCHEDULING AND ROUTING OPTIMIZATION IN TIME-SENSITIVE NETWORKS

By

AKRAM BILAL OMAR AKRAM

Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfilment of the Requirements for the Degree of Doctor of Philosophy

October 2024

FK 2024 61

DEDICATION

To my parents, Omar Akram Qassim and Huda Saeed Fathi, for their endless love, sacrifices, and unwavering belief in me.

To my brothers, Zaid and Ahmed, and my sister, Zainab, for their constant encouragement and the joy they bring to my life.

To the memory of my grandparents, Akram Qassim Yahya, Saeed Fathi Mustafa, and Fadhila Ahmed Abdullah, whose legacy continues to inspire me every day.

To my beloved grandmother, Mahrousa Nayef Mohammed, for her wisdom, warmth, and unconditional love.

To those who have stood by me through every challenge, offering their kindness, support, and friendship, your love and presence have been my strength and solace.

To my friends and colleagues in Iraq and Malaysia, for their camaraderie and shared moments of growth.

To my committee members, for their insightful feedback, guidance, and unwavering support throughout this journey.

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirement for the degree of Doctor of Philosophy

JOINT SCHEDULING AND ROUTING OPTIMIZATION IN TIME-SENSITIVE NETWORKS

By

AKRAM BILAL OMAR AKRAM

October 2024

Chairman : Professor Ir. Ts. Nor Kamariah binti Noordin, PhD
Faculty : Engineering

The rise of new technological applications, particularly in the domains of automation and autonomous vehicles, necessitates deterministic and reliable network communication with real-time responsiveness. However, existing methods frequently prioritize scheduling Time-Triggered (TT) traffic at the expense of optimizing lower-priority traffic, such as Best-Effort (BE), and often neglect joint routing considerations. Furthermore, the critical aspect of reliability in TT transmissions is often overlooked. This need has spurred the development and adoption of Time-Sensitive Networking (TSN), a set of evolving IEEE standards aiming to transform standard Ethernet for the demands of time-critical applications.

This work introduces two novel approaches to address these limitations: Optimized Hybrid Deterministic Scheduling and Routing (OHDSR) and its enhanced version, OHDSR+. Both methods prioritize communication efficiency and jointly optimize scheduling and routing for TT traffic while accommodating the requirements of BE traffic. While OHDSR ensures timely delivery of both TT and BE communications,

OHDSR+ extends this capability by guaranteeing the reliability of TT transmissions through redundant paths and time shift mechanisms.

In the experimental setup, a CP-Sat Solver from Google's OR-Tools was utilized for optimization. This was implemented using Python 3.11 within the Visual Studio Code environment. A diverse dataset was generated via a custom Python algorithm that leveraged the NetworkX library to create realistic network topologies. Problem instances were encoded in Extensible Markup Language (XML) for seamless integration. The dataset varied in size and complexity, encompassing diverse parameters such as streams, bridges, end-systems, applications, and tasks. Experiments were conducted on a standardized platform featuring an Intel Core i7-11370H CPU (3.30 GHz) with 8GB RAM, ensuring consistency and reliability in performance evaluations.

The OHDSR and OHDSR+ frameworks incorporate a gate control mechanism to prioritize TT traffic and manage gate closing times, preventing excessive delays for BE traffic. This results in stable worst-case latency for both traffic types. Furthermore, OHDSR+ demonstrates superior efficiency in handling higher redundancy levels, maintaining performance even under extreme redundancy scenarios. The detailed analysis of its performance within TSN reveals significant improvements in scheduling and routing optimization compared to existing methods, particularly in larger and more complex network topologies.

OHDSR excels in minimizing total latency while guaranteeing acceptable worst-case latency for both TT and BE traffic. Its scalability across various network sizes is

noteworthy, significantly outperforming similar approaches in terms of response times. Building upon this foundation, OHDSR+ further enhances network reliability by incorporating redundant paths for TT traffic transmission and implementing a time shift mechanism to ensure temporal diversity. This effectively mitigates the impact of potential failures and ensures the continuous flow of critical data.

Keywords: Time-Sensitive Networking (TSN); Joint Scheduling and Routing; Reliability; Real-Time Communication; Constraints Programming (CP)

SDG: GOAL 9: Industry, Innovation and Infrastructure

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia Sebagai memenuhi keperluan untuk ijazah Doktor Falsafah

PENJADUALAN BERSAMA DAN PENGOPTIMUMAN PENGHALAAN DALAM RANGKAIAN SENSITIF MASA

Oleh

AKRAM BILAL OMAR AKRAM

Oktober 2024

Pengerusi : Profesor Ir. Ts. Nor Kamariah binti Noordin, PhD
Fakulti : Kejuruteraan

Kemunculan aplikasi teknologi baharu, terutamanya dalam domain automasi dan kenderaan autonomi, memerlukan komunikasi rangkaian yang deterministik dan boleh dipercayai dengan respons masa nyata. Walau bagaimanapun, kaedah sedia ada sering mengutamakan penjadualan trafik Time-Triggered (TT) dengan mengorbankan pengoptimuman trafik keutamaan rendah, seperti Best-Effort (BE), dan sering mengabaikan pertimbangan penghalaan bersama. Tambahan pula, aspek kritikal kebolehpercayaan dalam penghantaran TT sering diabaikan. Keperluan ini telah mendorong pembangunan dan penerimaan Time-Sensitive Networking (TSN), satu set piawaian IEEE yang berkembang yang bertujuan untuk mengubah Ethernet standard untuk memenuhi keperluan aplikasi kritikal masa.

Kajian ini memperkenalkan dua pendekatan baharu untuk menangani batasan ini: Optimized Hybrid Deterministic Scheduling and Routing (OHDSR) dan versi terbaiknya, OHDSR+. Kedua-dua kaedah ini mengutamakan kecekapan komunikasi dan mengoptimumkan penjadualan serta penghalaan untuk trafik TT sambil

memenuhi keperluan trafik BE. Manakala OHDSR memastikan penghantaran tepat masa untuk kedua-dua TT dan BE, OHDSR+ memperluaskan keupayaan ini dengan menjamin kebolehpercayaan penghantaran TT melalui laluan berlebihan dan mekanisme anjakan masa.

Dalam persediaan eksperimen, CP-Sat Solver daripada OR-Tools Google digunakan untuk pengoptimuman. Pelaksanaannya dilakukan menggunakan Python 3.11 dalam persekitaran Visual Studio Code. Set data pelbagai dijana melalui algoritma Python tersuai yang memanfaatkan pustaka NetworkX untuk mencipta topologi rangkaian yang realistik. Contoh masalah dikodkan dalam Extensible Markup Language (XML) untuk integrasi lancar. Set data ini berbeza dari segi saiz dan kerumitan, merangkumi parameter seperti aliran, jambatan, sistem akhir, aplikasi, dan tugas. Eksperimen dijalankan pada platform standard dengan CPU Intel Core i7-11370H (3.30 GHz) dan 8GB RAM, memastikan konsistensi dan kebolehpercayaan dalam penilaian prestasi.

Kerangka OHDSR dan OHDSR+ menggabungkan mekanisme kawalan pintu untuk mengutamakan trafik TT dan mengurus masa penutupan pintu, mengelakkan kelewatan berlebihan untuk trafik BE. Ini menghasilkan kependaman kes paling buruk yang stabil untuk kedua-dua jenis trafik. Selain itu, OHDSR+ menunjukkan kecekapan yang lebih tinggi dalam mengurus tahap lebihan yang lebih tinggi, mengekalkan prestasi walaupun dalam senario lebihan yang melampau. Analisis terperinci prestasinya dalam TSN mendedahkan peningkatan ketara dalam pengoptimuman penjadualan dan penghalaan berbanding kaedah sedia ada, terutamanya dalam topologi rangkaian yang lebih besar dan kompleks.

OHDSR cemerlang dalam mengurangkan jumlah kependaman sambil menjamin kependaman kes paling buruk yang boleh diterima untuk kedua-dua trafik TT dan BE. Kebolehskalaannya merentasi pelbagai saiz rangkaian amat memberangsangkan, mengatasi pendekatan serupa dari segi masa respons. Berdasarkan asas ini, OHDSR+ meningkatkan lagi kebolehpercayaan rangkaian dengan menggabungkan laluan berlebihan untuk penghantaran trafik TT dan melaksanakan mekanisme anjakan masa untuk memastikan kepelbagaian temporal. Ini secara berkesan mengurangkan kesan kegagalan yang berpotensi dan memastikan aliran data kritikal yang berterusan.

Kata Kunci: Time-Sensitive Networking (TSN); Penjadualan Bersama dan Pengoptimunan Penghalaan; Kebolehpercayaan; Komunikasi Masa Nyata; Constraints Programming (CP)

SDG: MATLAMAT 9: Industri, Inovasi dan Infrastruktur

ACKNOWLEDGEMENTS

Alhamdulillah, all praise and gratitude are due to Allah, the Most Gracious, the Most Merciful, for granting me the strength, perseverance, and guidance to complete this thesis. I also would like to express my deepest gratitude to my supervisor, Prof. Dr. Nor Kamariah Noordin, for her invaluable support, encouragement, and mentorship throughout this journey. Her insightful feedback, patience, and unwavering belief in my abilities have been instrumental in shaping this thesis.

I am sincerely grateful to my thesis committee members for their constructive criticism, valuable insights, and time dedicated to reviewing my work. A special thank you to Prof. Dr. Mohd Fadlee A Rasid for sharing his vast knowledge and experience, which significantly enriched my understanding of the research area. His guidance and insightful questions helped me refine my arguments and strengthen the overall quality of this thesis. I would also like to thank Dr. Fazirulhisyam Hashim and Dr. Mustafa Ismail Salman for their insightful feedback and contributions.

I would also like to acknowledge the Sultan Abdul Samad Library, for providing me with the necessary resources and a conducive environment for my research. My sincere thanks also go to the Department of Computer and Communication Systems Engineering, particularly for their support and for fostering a collaborative and innovative atmosphere.

This journey would not have been possible without the unwavering love and support of my family. To my parents, Omar Akram Qassim and Huda Saeed Fathi, thank you for your endless sacrifices, unwavering belief, and constant encouragement. To my

siblings, Zaid, Ahmed, and Zainab, your love and support have been my refuge and strength throughout this challenging yet rewarding process. I would also like to thank my friend, Dema Mohammed Sabah, for her constant support and encouragement.

Finally, I want to express my sincere appreciation to everyone who has contributed to this thesis in ways big and small. My gratitude goes out to my friends and colleagues for their support and insightful discussions. Your contributions were invaluable to this journey.

This thesis was submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfilment of the requirement for the degree of Doctor of Philosophy. The members of the Supervisory Committee were as follows:

Nor Kamariah binti Noordin, PhD
Professor, Ir, Ts.
Faculty of Engineering
Universiti Putra Malaysia
(Chairman)

Fazirulhisyam bin Hashim, PhD
Associate Professor
Faculty of Engineering
Universiti Putra Malaysia
(Member)

Mohd Fadlee bin A Rasid, PhD
Professor
Faculty of Engineering
Universiti Putra Malaysia
(Member)

Mustafa Ismail Salman, PhD
Assistant Professor
Faculty of Engineering
University of Baghdad
(Member)

ZALILAH MOHD SHARIFF, PhD
Professor and Dean
School of Graduate Studies
Universiti Putra Malaysia

Date: 13 March 2025

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASAP	As Soon As Possible.
ATS	Asynchronous Traffic Shaper.
AVB	Audio Video Bridging.
BE	Best Effort.
CBS	Credit-Based Shaper.
CP	Constraint Programming.
CQF	Cyclic Queuing and Forwarding.
FIFO	First-In-First-Out.
FRER	Frame Replication and Elimination for Reliability.
GA	Genetic Algorithm.
Gbps	Giga bits per second
GCL	Gate Control List.
gPTP	generalized Precision Time Protocol.
GRASP	Greedy Randomized Adaptive Search Procedure.
IEEE	Institute of Electrical and Electronic Engineers.
ILP	Integer Linear Programming.
LAN	Local Area Network
OHDSR	Optimized Hybrid Deterministic Scheduling and Routing
OMT	Optimization Modulo Theories.
PBO	Pseudo-Boolean Optimization.
PSFP	Per-Stream Filtering and Policing.
QoS	Quality of Service.
QGT	Queue-Gating Time
SMT	Satisfiability Modulo Theories.
SRP	Stream Reservation Protocol.

TAS	Time-Aware Shaper.
TG	Task Group
TSN	Time-Sensitive Networking.
TT	Time-Triggered.
VLAN	Virtual LAN.
WCD	Worst-Case Delay
WCL	Worst-Case Latency
WG	Working Group

LIST OF SYMBOLS

HP	Hyperperiod
$\mathcal{G} =< \mathcal{V}, \mathcal{E} >$	Network graph
$\upsilon_a \in \mathcal{V}$	Node
$es_m \in ES$	End-System
$br_n \in BR$	Bridge
$q_{\phi}^{br_n} \in Q^{br_n}$	Queue number ϕ of the bridge br_n
$\rho_{\phi}^{q^{br_n}}$	Queue type (TT or BE) of the bridge br_n
$t_{\varphi}^{br_n} \in \Phi^{br_n}$	Gating time number φ of the bridge br_n
$g_q^{br_n} \in G_q^{br_n}$	Gate of the Queue q at the bridge br_n
$\psi_g^{br_n}$	Gate Status (1 or 0) of the Queue g at the bridge br_n
$\kappa_{\varphi}^{q^{br_n}}\boxed{\psi}$	Queue-Gating Time (QGT) for each queue q at status ψ
$(\upsilon_a, \upsilon_b) \in \mathcal{E}$	Edge (Link)
$\Omega_{(\upsilon_a,\upsilon_b)}$	Link Transmission Speed
$\delta_k \in \Delta$	Application
Γ_k^{δ}	Application Period
$\tau_j \in T$	Task
es_j^{τ}	Execution end-system
ω_j^{τ}	Worst-case execution time
Γ_j^{τ}	Task Period
$\eta_{\upsilon_a}^{\tau}$	Task offset at node υ_a
$\zeta_{\upsilon_a}^{\tau}$	Task end-time at node υ_a
$\sigma_i \in S$	Stream
L_i^{σ}	Stream Size

Γ_i^σ	Stream period
ρ_i^σ	Stream type (TT or BE)
Π_i^σ	Stream redundancy status
$\eta_{(\upsilon_a,\upsilon_b)}^\sigma$	Stream offset at link (υ_a, υ_b)
$\varepsilon_{(\upsilon_a,\upsilon_b)}^\sigma$	Stream duration at link (υ_a, υ_b)
$\zeta_{(\upsilon_a,\upsilon_b)}^\sigma$	Stream end-time at link (υ_a, υ_b)
$\tau_{talker}^{\sigma_i}$	Source task
$T_{listener}^{\sigma_i}$	Destination tasks
$\mathbb{R}_i^\sigma$	Stream route

CHAPTER 1

INTRODUCTION

The transition towards the era of real-time networks has brought about an exponential surge in demand for reliable, ultra-low latency communication solutions. Time-Sensitive Networking (TSN), a promising technology comprising a set of standards developed by the Institute of Electrical and Electronics Engineers Standards Association (IEEE SA) [1], has emerged as a key player in addressing this demand. Research in the TSN field is rapidly expanding, with a particular emphasis on scheduling techniques for critical communication. This research considers multiple facets, including reliability, to achieve low-latency communication with minimal data loss. This chapter delves into the background of TSN and explores the challenges inherent in its implementation.

The chapter begins by highlighting the core concepts and objectives of TSN, providing a foundation for understanding its role in enabling real-time communication. It then examines the specific problems and challenges encountered within TSN, such as scheduling complexity, synchronization requirements, and ensuring reliability in the face of potential disruptions.

Following this, the chapter outlines the main research objectives and scope of this thesis, defining the specific areas of investigation and the intended contributions to the field of TSN. This includes the development of novel scheduling and routing algorithms, the exploration of redundancy mechanisms for enhanced reliability, and the consideration of factors such as network scalability and resource management. Finally, the chapter provides a clear outline of the thesis organization, guiding the

reader through the subsequent chapters and their respective contributions to the overall research objectives. By establishing a comprehensive understanding of TSN, its challenges, and the research goals, this chapter sets the stage for a detailed exploration of the proposed solutions and their impact on the future of real-time communication networks.

1.1 Background and Motivation

The recent expansion in networking and its integration with new industries has led to the development of complex networks that are not adequately equipped for real-time communications. Network protocols must excel in scalability, cost, compatibility, safety, and real-time application handling. Although Ethernet [2] is a well-established network protocol, it is not well-suited for real-time applications [3]. In recent years, these features have become critical for modern networks and applications, including those used in automation industries and autonomous vehicles. To address these challenges, various Ethernet-based protocols have been proposed, including Time-Triggered (TT) Ethernet and Audio/Video Bridging (AVB). However, none of these later protocols fully meet the requirements of new real-time applications [4]. A new technology, Time-Sensitive Networking (TSN), was introduced, building on TT-Ethernet and AVB to offer enhanced real-time features for Ethernet networks [5].

Building on various extensions related to TT-Ethernet and AVB, the Institute of Electrical and Electronics Engineers Standards Association (IEEE SA) [1] initiated the IEEE 802.1 Time-Sensitive Networking (TSN) Task Group (TG). This group has established a set of standards that enhance Ethernet capabilities. These standards introduce new features that bolster the robustness and determinism of the networks.

Additionally, they enable real-time synchronization and allow for effective management of different traffic types [6]. The TSN framework is designed to handle hybrid traffic encompassing three types, each with distinct priorities for mixed-criticality applications [7]. The highest priority class, which requires bounded latency and zero-jitter, is the TT traffic. This is articulated in IEEE 802.1Qbv [8]. The mechanism employed to configure the TT traffic in the Gate Control List (GCL) schedule is known as Time-Aware Shaping (TAS). While TAS manages the highest-priority class, the Credit-Based Shaping (CBS) mechanism configures the subsequent highest traffic class: the AVB traffic. CBS was introduced in IEEE 802.1Qav [9]. The final TSN traffic class is Best-Effort (BE). Since there is no need for a timing guarantee for this traffic type, it is regarded as the lowest priority class.

1.2 Problem Statements

The need for real-time networks has significantly increased in recent years, driven primarily by the emergence of technologies within the industrial automation domain. Time-Sensitive Networking (TSN) has emerged as a promising solution for these networks. However, achieving a reliable, well-managed network with deterministic low latency remains an ongoing challenge for TSN, crucial for meeting the Quality of Service (QoS) demands of real-time applications.

- Efficient scheduling and routing time-triggered flows within Time-Sensitive Networks (TSN) presents a significant challenge due to the inherent complexity of the problem, classified as NP-hard. Traditional approaches often result in suboptimal latency for Time-Triggered (TT) traffic, especially when dealing with increased network load, complex topologies, or a high number of concurrent high-priority flows. This congestion leads to increased latency and potential frame loss, ultimately limiting the scalability and

schedulability of TSN for time-critical applications where minimizing latency and ensuring reliable delivery are crucial.

- While recent research has primarily focused on scheduling Time-Triggered (TT) traffic in Time-Sensitive Networks (TSN), the management of Best-Effort (BE) traffic, often considered low-priority, requires further attention. Both TT and BE traffic compete for the same network resources, and without proper optimization, the performance and schedulability of BE traffic can suffer significantly. This highlights the need for scheduling approaches that effectively balance the requirements of both traffic types to ensure efficient and fair resource allocation within TSN.
- In Time-Sensitive Networks (TSN), ensuring the reliability of Time-Triggered (TT) communications is paramount, particularly in safety-critical applications like industrial control systems where data loss or delays can have severe consequences. While current TSN redundancy mechanisms exist, they may be inadequate or overly complex to implement effectively. Therefore, developing robust and efficient methods for reliable routing and scheduling of TT traffic remains a critical challenge in TSN to guarantee the successful delivery of time-sensitive data.

1.3 Research Objectives

In light of the problems outlined in the previous section, the following key objectives have been established to address these challenges:

i. To propose and develop the Optimized Hybrid Deterministic Scheduling and Routing (OHDSR) framework that handles joint scheduling and routing for TT traffic, minimizing response time, achieving low-latency transmission, and improving the scalability of TSN. This objective includes proposing the application of the Queue-Gating Time (QGT) approach for bridge management and data flow optimization.

ii. To optimize the scheduling and routing of BE traffic within the OHDSR framework to increase the overall schedulability and handling of low priority traffic, ensuring balanced network performance.

iii. To enhance the reliability of TT traffic through the development of the OHDSR+ model, which incorporates both temporal-based and spatial-based redundancy mechanisms. These mechanisms use redundant paths, and time shifts to ensure robust communication, minimizing data loss and maintaining reliable data delivery under network disruptions.

1.4 Research Scope

The rise of real-time applications across industries, such as industrial automation, automotive systems, and professional audio/video streaming, drives the need for reliable, deterministic communication networks. Time-Sensitive Networking (TSN) emerges as a transformative technology, providing standards and mechanisms designed for this purpose over standard Ethernet. While legacy technologies like CAN, FlexRay, AFDX, EtherCAT, PROFINET, and Sercos III addressed real-time needs in their domains, TSN offers a more versatile and scalable solution for a wider range of applications.

TSN standards cover precise time synchronization, traffic scheduling and shaping, network management, and robust reliability mechanisms. This thesis focuses on two critical areas: scheduling, and reliability, with an emphasis on Time-Triggered (TT) traffic, the highest priority class in TSN. Acknowledging the impact of lower-priority Best-Effort (BE) traffic, our work examines TT traffic transmission. The Time-Aware Shaper (TAS) is crucial for shaping and prioritizing traffic flows, ensuring that time-sensitive data meets its strict timing requirements.

Optimization techniques determine the best routes and schedules for TT and BE traffic, balancing determinism with network resource efficiency. Bridge design is fundamental to our research, particularly the configuration of queues and gates within TSN switches. Governed by a Gate-Control List (GCL), these elements orchestrate data flow based on priorities and timing. This control minimizes delays and jitters for TT traffic, supporting real-time applications.

Reliable communication for TT traffic is paramount. This thesis investigates temporal and spatial redundancy mechanisms to mitigate network failures. Temporal redundancy involves transmitting duplicate packets at different times, while spatial redundancy uses multiple paths, ensuring critical information reaches its destination even if one path fails. The study leverages Google's OR-Tools CP-Sat Solver for optimization, implemented in Python 3.11 within Visual Studio Code. Experiments are conducted on a standardized platform featuring an Intel Core i7-11370H CPU and 8GB RAM.

By combining optimized scheduling and routing with robust redundancy, TSN provides a reliable and deterministic foundation for many time-sensitive applications. Figure 1.1 illustrates the scope of the thesis. The yellow and green boxes indicate the methods and protocols covered, respectively. The orange boxes represent the performance parameters, along with the concept of optimization, which is also depicted in orange. The blue boxes represent the traffic types covered in the thesis. Other techniques and protocols that are not within the scope of this research are depicted in gray.

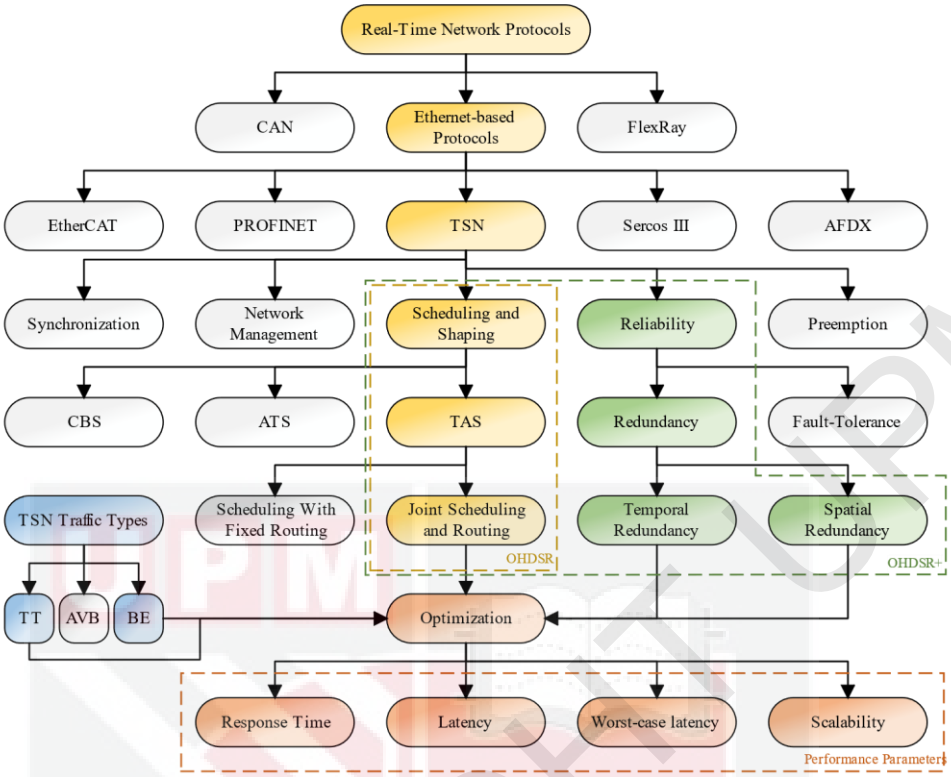

Figure 1.1 : Study Scope

1.5 Research Contributions

The key contributions of our thesis are summarized as follows:

1. Optimized Hybrid Deterministic Scheduling and Routing (OHDSR): This novel model tackles the joint routing and scheduling problem for Time-Triggered (TT) and Best-Effort (BE) traffic within Time-Sensitive Networks (TSN). By incorporating a priority-based stream constraint that governs the gating mechanism at each bridge's egress port, OHDSR ensures efficient and prioritized handling of diverse traffic types.

2. Innovative Queue-Gating Time (QGT) Bridge Model: A unique bridge model employing the Queue-Gating Time (QGT) approach is introduced. This method effectively manages queueing for various traffic types and dynamically determines gate opening and closing times, thereby optimizing data flow and network resource utilization.

3. OHDSR+ with Enhanced Reliability for Time-Triggered Traffic: Building upon the OHDSR framework, the Optimized Hybrid Deterministic Scheduling and Routing Plus (OHDSR+) model is presented. This enhanced model prioritizes the reliability of TT communications by incorporating redundant paths for data transmission and implementing a time shift mechanism. The time shift ensures temporal diversity between the original and redundant traffic, significantly reducing the risk of data loss and guaranteeing reliable delivery even in the presence of network disruptions.

1.6 Thesis Organization

The structure of this thesis is as follows:

This chapter begins with a brief overview of Time-Sensitive Networking (TSN), setting the context for the thesis. It then articulates the research problem, identifying the specific challenges this work addresses. Next, the chapter presents the research objectives, outlining the key aims of the study. The research scope is defined, marking the boundaries of the investigation and focusing on specific TSN aspects. Following this, the chapter details the study's contributions, highlighting the novel insights and advancements achieved. Lastly, the chapter provides a roadmap of the thesis, guiding the reader through the subsequent chapters.

Chapter Two, Literature Review delves into the background and existing body of knowledge surrounding Time-Sensitive Networking (TSN). It begins by exploring the architecture and standards of TSN, including key mechanisms like traffic shaping (TAS, CBS, ATS) and reliability protocols. The chapter then examines scheduling approaches within TSN, highlighting the complexity of the problem and various solution strategies. A particular focus is placed on joint scheduling and routing (JSR)

with considerations for multicast traffic, task scheduling, reliability, and non-critical traffic management.

Chapter Three, Research Methodology outlines the methodology employed in this research. It introduces the system model, encompassing both the network and application models, and details the innovative Queue-Gating Time (QGT) approach for bridge management. Exemplary problem instances are presented, including formulations for unicast/multicast scenarios and considerations for redundancy. The chapter further explains the optimization framework and the various constraints applied, encompassing routing, priority scheduling, stream scheduling, task scheduling, and stream schedule reliability. Finally, the objective functions used for evaluation are defined.

Chapter Four, Results and Discussion presents the experimental results obtained through the evaluation of the proposed OHDSR and OHDSR+ approaches. The setup and problem instances are described, followed by a thorough analysis of performance metrics such as latency, response time, and scalability. Comparisons are drawn between the two models, highlighting the advantages of OHDSR+ in terms of reliability and efficiency.

This final Chapter, Conclusion and Future Work summarizes the key findings of the research and reiterates the contributions of the OHDSR and OHDSR+ models to the field of TSN scheduling and routing. Additionally, potential avenues for future research and development are explored, paving the way for further advancements in TSN technology.

CHAPTER 2

LITERATURE REVIEW

This chapter provides a comprehensive overview of Time-Sensitive Networking (TSN), exploring its architecture, standards, and key mechanisms such as traffic shaping and reliability protocols. It examines various scheduling approaches within TSN, highlighting the complexity of the scheduling problem and exploring diverse solution strategies. Particular emphasis is placed on joint scheduling and routing (JSR) optimization, considering factors like multicast traffic, task scheduling, reliability, and non-critical traffic management.

2.1 Introduction

The expansion of emerging technologies such as autonomous vehicles, avionics systems, and advanced industrial automation has heightened the need for robust real-time communication capabilities [10]. These applications often involve mission-critical tasks where low-latency data transmission and absolute reliability are paramount. Consequently, the underlying communication networks must exhibit precise timing characteristics to ensure minimal delays and zero data loss.

While Ethernet [2] has long been the dominant networking technology due to its ease of deployment and robust performance, its inherent lack of support for real-time communication presented a significant challenge. This deficiency led to the development of application-specific protocols such as Controller Area Network (CAN) and FlexRay for automotive systems, Avionics Full-Duplex Switched Ethernet (AFDX) for avionics applications, and EtherCAT, PROFINET, and Sercos III for

industrial automation. However, these disparate protocols often suffer from compatibility issues when integrated with Ethernet-centric networks, limiting their interoperability and adaptability across diverse industries [11].

Recognizing the limitations of existing solutions, the IEEE Time-Sensitive Networking (TSN) task group has actively pursued the standardization of technology that enables deterministic communication over Ethernet. TSN augments standard Ethernet capabilities by guaranteeing the delivery of high-priority traffic within precisely defined timeframes. This deterministic nature ensures that critical data packets reach their destination on time, every time, making TSN an ideal solution for addressing the stringent demands of real-time communication in various emerging technological domains.

2.2 Background on TSN

Standard Ethernet, while ubiquitous, lacks the deterministic real-time capabilities demanded by emerging applications in autonomous vehicles and industrial automation [10], [11]. To address this, the IEEE 802.1 Time-Sensitive Networking (TSN) task group developed a suite of standards enhancing Ethernet with real-time features [12]. Initially based on the foundational work of Audio Video Bridging (AVB) [13], [14], TSN evolved to overcome AVB's limitations in scalability and preemptive scheduling, providing improved mechanisms for guaranteed data delivery and low latency. This enabled deterministic communication crucial for the reliable operation of Industry 4.0 applications [15], providing precise timing and guaranteed data delivery for mission-critical tasks within industrial control systems. This deterministic behavior is achieved through features such as time-aware scheduling and traffic shaping, ensuring precise

timing and guaranteed data delivery. The evolution from AVB to TSN reflects a broader recognition of the need for deterministic communication across diverse industries, where the consequences of timing failures can be significant. The following sections will delve into the specifics of TSN traffic classes, scheduling mechanisms, and traffic shaping techniques, examining how these features contribute to the overall reliability and performance of TSN networks, and how they address the challenges inherent in real-time communication.

2.2.1 TSN Standards and Mechanisms

TSN significantly enhances standard Ethernet by integrating a suite of features that ensure reliable, low-latency communication crucial for time-critical applications in sectors like industrial automation, automotive, and avionics. Table 2.1 summarizes the published TSN standards, categorizing them by functionality such as time synchronization, low latency, high availability, resource management, and application-specific implementations.

Building upon the foundation of AVB, TSN addresses the inherent limitations of traditional Ethernet through precise time synchronization, deterministic scheduling, and robust reliability mechanisms [16]. Precise time synchronization, achieved via the Generalized Precision Time Protocol (gPTP), based on the industry-standard Precision Time Protocol (PTP, IEEE 1588 [17]), and defined in IEEE 802.1AS [18], enables sub-microsecond accuracy using a Grandmaster clock [5]. This shared time base ensures coordinated actions across the network and promotes predictable network behavior, essential for time-sensitive operations [19], [20].

Table 2.1 : List of TSN Published Standards

Standards categories	Standard	Title	Publication date
Time synchronization	802.1AS [18]	Timing and Synchronization	2020
Low latency	802.1Qav [9]	Credit Based Shaper	2010
	802.1Qbu [21]	Frame Preemption	2016
	802.1Qbv [8]	Scheduled Traffic	2015
	802.1Qch [22]	Cyclic Queuing and Forwarding	2017
	802.1Qcr [23]	Asynchronous Traffic Shaping	2020
High availability / Ultra reliability	802.1CB [24]	Frame Replication and Elimination	2017
	802.1Qca [25]	Path Control and Reservation	2016
	802.1Qci [26]	Per-Stream Filtering and Policing	2017
Dedicated resources & API	802.1Qat [27]	Stream Reservation Protocol	2010
	802.1CS [28]	Link-local Registration Protocol	2021
	802.1Qcc [29]	Stream Reservation Protocol (SRP) Enhancements and Performance Improvements	2018
	802.1Qcp [30]	YANG Data Model	2018
	802.1Qcx [31]	YANG Data Model for Connectivity Fault Management	2020
Application-Specific TSN Standards	802.1BA [32], [33], [34]	Audio Video Bridging (AVB) Systems	2011, 2016, 2021
	802.1CM [35] / 802.1CMde [36]	TSN for Fronthaul	2018, 2020

To manage traffic flow and prioritize critical data, TSN employs various traffic shaping and scheduling techniques. These include the Time-Aware Shaper (TAS, IEEE 802.1Qbv [8]), which implements time-triggered scheduling [37]; the Credit-Based Shaper (CBS, IEEE 802.1Qav [9]), suitable for audio/video streams; and Asynchronous Traffic Shaping (ATS, IEEE 802.1Qcr [23]), which offers flexibility for various traffic patterns. Frame preemption (IEEE 802.1Qbu [21]) further enhances timely delivery by allowing higher-priority frames to interrupt lower-priority

transmissions, minimizing latency for critical data. Reliability is significantly bolstered through mechanisms like Frame Replication and Elimination for Reliability (FRER, IEEE 802.1CB [24]), which transmits redundant frames over independent paths to mitigate the impact of link failures. Path Control and Reservation (IEEE 802.1Qca [25]) provides resource reservation for critical traffic, ensuring guaranteed bandwidth and minimal delay.

Further refinements to TSN's capabilities are provided by the Stream Reservation Protocol (SRP, IEEE 802.1Qat [27]) and its subsequent enhancements (IEEE 802.1Qcc [29] and IEEE 802.1Qcp [30]), which guarantee bandwidth and priority allocation for specific streams. Per-stream filtering and policing (IEEE 802.1Qci [26]) and cyclic queuing and forwarding (IEEE 802.1Qch [22]) contribute to refined performance by preventing congestion and unauthorized access, ensuring consistent and predictable network behavior. Application-specific standards, such as Audio Video Bridging Systems (IEEE 802.1BA [34]) and TSN for Fronthaul (IEEE 802.1CM [35]), further demonstrate TSN's adaptability to diverse use cases. These standards enable standard Ethernet to meet stringent real-time demands by providing low latency, and high reliability.

2.2.2 TSN Traffic Shaping

Effective traffic shaping is crucial for TSN to guarantee the timely delivery of time-critical data while efficiently utilizing network resources. TSN employs several key mechanisms to achieve this, including Time-Aware Shaping (TAS), Credit-Based Shaping (CBS), and Asynchronous Traffic Shaping (ATS). TAS [8], provides deterministic scheduling, guaranteeing bounded latency and jitter for time-critical

traffic by allocating dedicated time slots within a repeating cycle. This makes it ideal for applications requiring precise timing, such as industrial control systems. CBS [9], regulates data flow using a credit system, suitable for less time-sensitive traffic (e.g., AVB, BE). It prevents network overload and ensures fair bandwidth allocation without strict scheduling. Figure 2.1 shows the integration of TAS and CBS within an 802.1Qbv-enabled bridge.

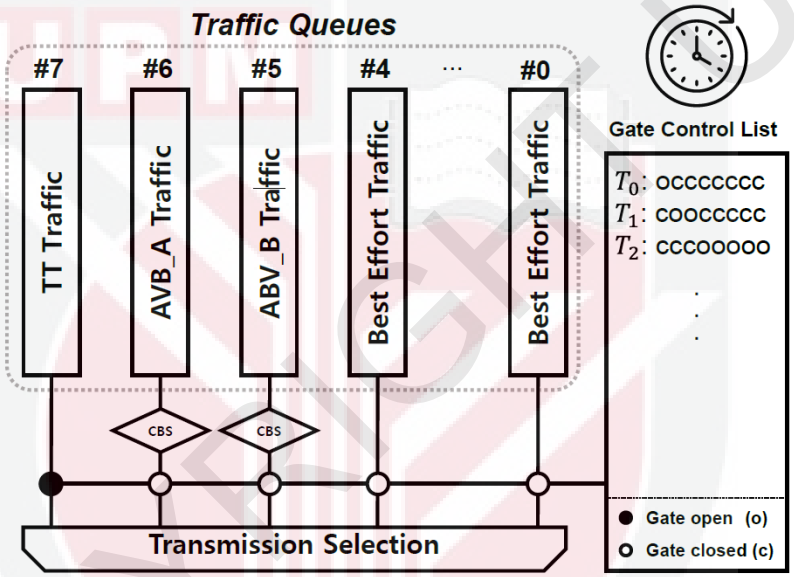

Figure 2.1 : Time-Aware Scheduling (TAS) and Credit-Based Shaper (CBS) in an 802.1Qbv-Enabled Bridge [38]

Asynchronous Traffic Shaping (ATS, IEEE 802.1Qcr [23]) offers an alternative for shaping time-triggered traffic where the cyclic nature of TAS is less suitable, such as applications with sporadic, bursty traffic. These techniques, summarized in Table 2.2, are essential for building efficient and reliable time-sensitive networks. While other techniques like burst limiting shapers (BLS) and peristaltic shapers (PS) exist, TAS, CBS, and ATS have gained wider adoption within TSN standards due to their effectiveness in diverse applications.

Table 2.2 : Comparison of TSN Shaping Techniques

Shaping Technique	Scheduling	Latency	Suitability
TAS	Deterministic	Guaranteed Bounded	Time-critical (TT)
CBS	Non- Deterministic	Not Guaranteed	Less time-critical (AVB, BE)
ATS	Asynchronous	Guaranteed Bounded	Sporadic, bursty TT

2.2.3 TSN Traffic Classes

Real-time applications, particularly within the domains of automation and automotive systems, present a diverse range of requirements and demands on the underlying network infrastructure. TSN addresses this heterogeneity by organizing traffic into distinct classes based on priority and criticality, enabling efficient management of transmissions according to the specific needs of each class. TSN defines three primary traffic classes, each with distinct QoS requirements and prioritized as follows [39], [40]:

- Time-Triggered (TT): This class encompasses the highest priority traffic, characterized by stringent requirements for bounded end-to-end latency and minimal jitter. TT traffic typically involves critical control messages or time-sensitive data that demand precise and predictable delivery times.
- Audio-Video Bridging (AVB): Occupying a mid-high priority level, AVB traffic requires a guaranteed latency bound to ensure seamless and uninterrupted delivery of multimedia streams. While not as critical as TT traffic, AVB still necessitates timely delivery to maintain the quality and integrity of audio and video content.
- Best-Effort (BE): This class represents the lowest priority traffic with no specific timing requirements. BE traffic typically includes non-critical data that can tolerate some degree of delay or variability in delivery times.

TSN effectively manages diverse traffic with varying timing requirements through a hierarchical classification scheme. This prioritizes time-critical data while accommodating less sensitive traffic. TSN categorizes traffic into three main classes: Time-Triggered (TT), Audio Video Bridging (AVB), and Best-Effort (BE). Figure 2.2 illustrates the hierarchical structure of TSN traffic classes.

TT traffic, with stringent timing constraints (e.g., control messages in automation systems [5]), is managed by the Time-Aware Shaper (TAS). TAS allocates dedicated time slots, ensuring timely delivery based on priority code points (PCPs) in 802.1Q frames [5], [12]. AVB traffic (audio/video streams) requires bounded latency, less stringent than TT. The Credit-Based Shaper (CBS) manages AVB, offering prioritized Class A and less strictly reserved Class B streams [40]. Finally, BE traffic (traditional Ethernet) receives no timing guarantees, using remaining bandwidth [41]. This prioritized approach ensures that time-critical applications receive necessary resources, maintaining determinism and reliability.

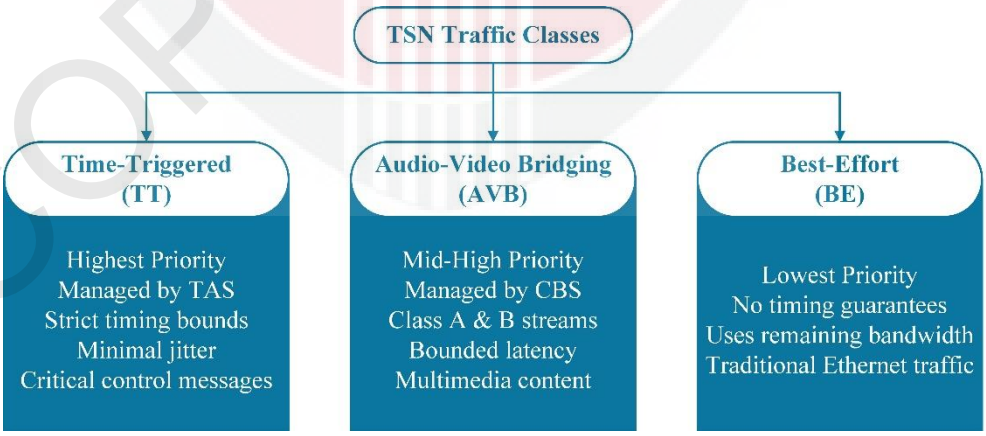

Figure 2.2 : Hierarchical Structure of TSN Traffic Classes

2.3 Time-Aware Shaping (TAS)

Time-Aware Shaping (TAS) is a crucial mechanism in TSN that ensures deterministic communication by controlling the transmission of time-critical data streams [38], [42]. Operating on the Time Division Multiple Access (TDMA) principle, TAS divides time into discrete slots, which are managed by Gate Control Lists (GCLs) [43], [44]. This allows the highest priority traffic, such as TT flows, to be allocated dedicated time slots, ensuring minimal latency and jitter. This precise scheduling is vital for applications demanding high reliability and predictability, such as industrial automation, automotive systems, and professional audio/video streaming [38], [45]. Synchronization across the network, achieved via the IEEE 802.1AS protocol, prevents timing conflicts and supports smooth traffic flow.

TAS also employs guard bands to protect critical traffic from interference caused by lower-priority data flows, further enhancing its deterministic nature. Despite its benefits, implementing TAS in large-scale networks presents several challenges. The process of generating optimized GCLs is NP-hard, especially in networks with numerous data flows, making scheduling complex [46], [47], [48]. Additionally, the fixed-schedule approach of TAS can lead to wasted bandwidth when time slots remain unused. This highlights the need for advanced scheduling algorithms that balance bandwidth efficiency with the timely delivery of TT traffic [49]. Figure 2.3 shows a TAS at a bridge's egress port, controlling queue transmission gates via a GCL.

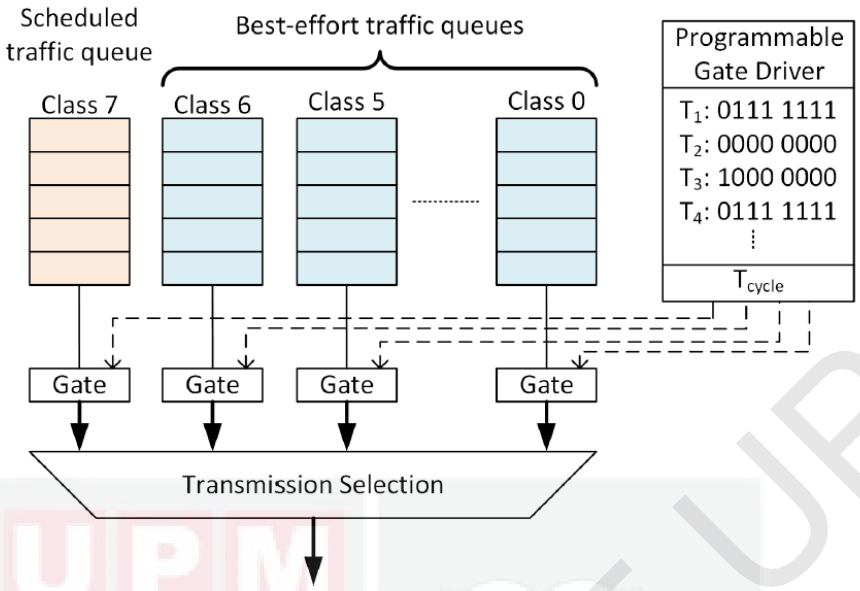

Figure 2.3 : Time-Aware Shaper (TAS) in an Egress Port of a Bridge Controls the Transmission Gates at Each Queue Using a Gate Control List (GCL) [20]

The goal of TAS scheduling is to configure GCLs across switches in the network to ensure punctual delivery of TT traffic while minimizing delays. Optimizing factors like the number of time slots and the size of guard bands is essential to achieve the balance between reliable performance and efficient resource utilization. However, as network topology and the number of TT flows increases, scheduling complexity escalates. Managing multiple traffic paths and dependencies, alongside maintaining precise time synchronization, becomes more challenging as the network grows in size.

Despite these challenges, research is ongoing to improve TAS implementation through advanced scheduling algorithms that address the complexities of large-scale networks. By optimizing bandwidth utilization and tackling the difficulties associated with network topology and synchronization, TAS can continue to be a vital tool for supporting the growing demands of real-time applications across diverse industries.

2.3.1 Bridges and Gate Control List (GCL)

TSN ensures deterministic communication through mechanisms like TAS, which plays a crucial role in managing data transmission within TSN bridges (switches). Figure 2.4 shows a simplified logical diagram of an 802.1Qbv-enabled bridge. TAS utilizes Gate Control Lists (GCLs) to schedule transmission queues, dividing time into specific intervals and assigning them to different traffic types. This time-based scheduling guarantees that time-sensitive traffic, like TT flows, is prioritized and isolated, ensuring timely delivery and preventing interference from lower-priority traffic.

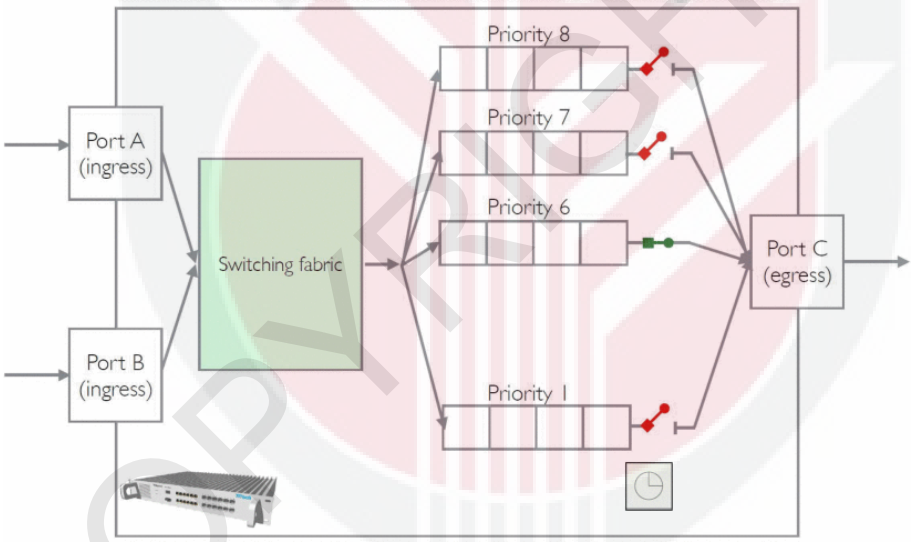

Figure 2.4 : Simplified View of an 802.1Qbv-Capable Bridge (Switch) [50]

The core of TAS lies in GCLs, which are static schedules that define the precise timing for opening and closing the gates of TSN switches [49]. GCLs can be tailored to enforce bandwidth reservation or time-based shaping, ensuring strict control over traffic flow. For TT traffic, GCLs can be configured to allow exclusive access to the transmission channel, eliminating interference and guaranteeing the timely delivery of

critical data. This high level of determinism is essential for real-time applications where predictable performance is crucial, such as industrial automation and automotive systems.

Precise time synchronization across the entire network is essential for TAS to function correctly, achieved through the IEEE 802.1AS protocol [51]. This synchronization ensures that all devices follow a unified time reference, adhering to the GCLs to open and close gates at designated times [49], [52]. The GCLs repeat cyclically, maintaining a predictable and consistent flow of traffic. To prevent delays from lower-priority traffic, guard bands—buffer zones placed between time slots—ensure that critical transmissions occur without intrusion, preserving the deterministic behavior of time-sensitive traffic [5].

The true innovation of TAS is its ability to isolate traffic flows through scheduled time slots, rather than simply relying on prioritization [53]. This approach ensures that high-priority Scheduled Traffic (ST) enjoys deterministic latency and jitter guarantees, making it ideal for time-critical control systems [16], [53]. Other traffic types, such as AVB or BE traffic, can be efficiently managed within the remaining time slots or according to their priority levels.

In summary, the combination of bridges, GCLs, and TAS establishes the foundation for deterministic communication in TSN. By carefully crafting GCLs and employing time-based scheduling, TSN ensures that time-sensitive data is reliably transmitted in real-time, unaffected by lower-priority traffic. This level of control is vital for

industries where predictable communication is essential for safety and smooth operation, including industrial automation, automotive systems, and more.

2.4 TSN Scheduling

Deterministic transmission of time-sensitive traffic is crucial for TSN, requiring bounded latency and jitter. This is achieved primarily through scheduling, particularly for TT traffic. Scheduling TT traffic within TSN is a challenging task requiring optimal solutions to maintain deterministic network performance. The main goal is to design a schedule with precise timing to deliver data traffic in a specific sequence across devices, meeting stringent QoS requirements. This ensures reliable transmission, prevents data loss, and supports time-sensitive applications.

Achieving an optimal solution involves addressing numerous factors, such as network topology, traffic characteristics, resource constraints, and timing requirements. The complexity lies in minimizing latency and jitter while meeting deadlines and efficiently utilizing bandwidth. Moreover, the schedule must be robust against potential disruptions to maintain reliable communication.

Effectively solving TSN scheduling is critical for unlocking TSN's full potential, enabling real-time applications across industries like industrial automation and automotive networking. This involves defining suitable variables, constraints, and objective functions. Variables may include transmission times, queue assignments, and path selections, while constraints ensure adherence to deadlines and network requirements.

By employing the right optimization approaches and tailoring problem parameters, researchers can tackle the complexities of TSN scheduling. This ensures deterministic and efficient communication, fostering advancements in time-sensitive systems across diverse domains.

2.4.1 Scheduling Complexity

As previously discussed, scheduling in TSN presents a complex problem demanding significant computational effort to find optimal solutions. Classifying the complexity of scheduling problems helps us understand the inherent challenges and guides our approach towards finding efficient solutions. One classification category is the P class, encompassing problems solvable in Polynomial Time (P). In these problems, a solution can be found within a time frame that grows polynomially with the size of the input. Another category, the Nondeterministic Polynomial time (NP) class, includes all problems where a potential solution can be verified in polynomial time. The P class problems form a subset of the NP class.

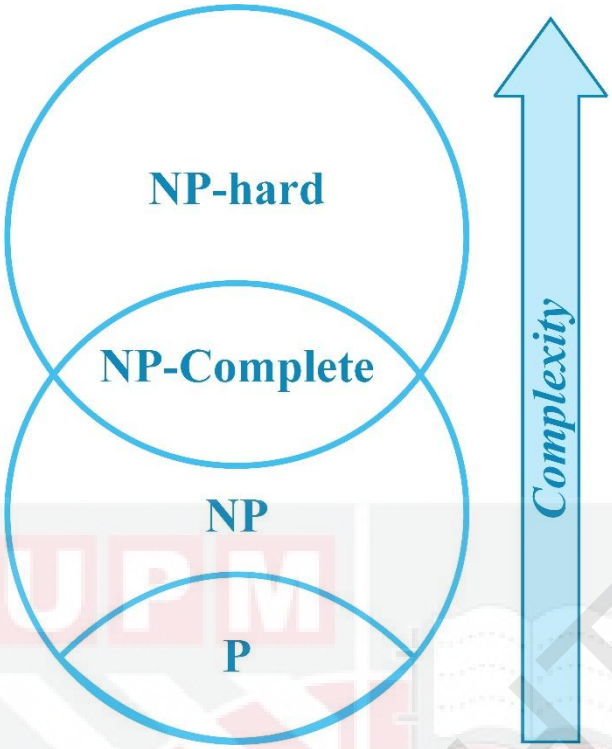

Figure 2.5 : Illustration of Various Complexity Classes

Within the NP class, problems that are considered "hardest" are categorized as NP-complete. These problems have the property that any other problem in the NP class can be reduced to them in polynomial time. In essence, solving an NP-complete problem efficiently would imply the ability to solve all NP problems efficiently. The challenge escalates further with NP-hard problems. This category encompasses NP-complete problems and even more complex problems that may not necessarily have solutions verifiable in polynomial time [54]. Figure 2.5 provides a graphical illustration of the relationships and hierarchical structure among different computational complexity classes

TSN scheduling problems predominantly fall under the NP-complete or NP-hard classifications. When a problem can be reduced to the well-known Bin Packing problem, where items of different sizes need to be packed into a finite number of bins

optimally, it is classified as NP-complete [55], [56]. Conversely, if a problem has the potential to be reduced to the Bin Packing problem, it falls under the NP-hard category as referenced in [46] and [57]. Understanding these complexity classifications is crucial for researchers and network engineers working with TSN. It helps set realistic expectations regarding the computational effort required for finding optimal solutions and guides the selection of appropriate algorithms and optimization techniques.

2.5 TSN Scheduling Approaches

Addressing the complexities of scheduling in Time-Sensitive Networking (TSN) requires a diverse set of approaches, each with its strengths and considerations. Satisfiability Modulo Theories (SMT) solvers offer a powerful tool for handling constraint satisfaction problems, ensuring that all timing, resource, and dependency requirements are met. Integer Linear Programming (ILP) excels at modeling and optimizing scheduling constraints within a mathematical framework, often leading to optimal or near-optimal solutions. Constraint Programming (CP) provides a flexible framework for expressing complex timing and dependency relationships between traffic flows, enabling efficient exploration of the solution space. Finally, heuristic approaches offer efficient solutions, albeit not always optimal, by leveraging problem-specific knowledge and search strategies. Research in TSN scheduling explores these diverse approaches, considering factors like reliability, low-priority traffic handling, and computational complexity to find the most effective methods for achieving deterministic and efficient communication in time-sensitive networks.

2.5.1 Satisfiability Modulo Theories (SMT)

At its core, satisfiability refers to the fundamental problem of determining whether a formula expressing a constraint possesses a model or solution [58]. Building upon this concept, Satisfiability Modulo Theories (SMT) extends the problem to encompass logical formulas over one or more first-order theories [59]. SMT solvers, specialized tools designed to tackle this decision problem, play a crucial role in various applications, including test case generation, predicate abstraction, extended static checking, and notably, solving scheduling problems within Time-Sensitive Networking (TSN).

Among the most widely adopted SMT solvers for TSN scheduling is the Z3 Solver [60]. This powerful tool excels at handling complex constraints and finding solutions that satisfy all specified conditions. By leveraging the capabilities of Z3, researchers and network engineers can effectively model and solve TSN scheduling problems, ensuring that timing deadlines are met, resources are allocated efficiently, and data transmission occurs reliably.

2.5.2 Integer Linear Programming (ILP)

An Integer Linear Program (ILP) provides a mathematical framework for describing the solution space of a problem using linear inequalities. Each assignment of variables that satisfies all inequalities corresponds to a feasible solution, and conversely, every solution can be represented by a set of variable assignments fulfilling the constraints. In certain cases, variables may be restricted to integer values. A linear objective function can be defined to evaluate the quality of solutions, where the objective is to minimize the function value.

ILP solvers are employed to compute a feasible assignment that minimizes the objective function. During the solution process, each encountered solution provides an upper bound on the objective value of the optimal solution. Moreover, the solver can infer lower bounds for the optimal objective value, offering valuable insights even when finding the exact optimal solution within a reasonable timeframe is not feasible. These bounds provide estimations of the maximum gap between the best-found solution and the true optimum. Widely adopted state-of-the-art ILP solvers include CPLEX [61] and Gurobi [62], which offer efficient algorithms for solving ILP problems.

2.5.3 Constraint Programming (CP)

Constraint Programming (CP) offers a versatile approach for tackling combinatorial problems by describing the set of feasible solutions in a declarative manner. In this regard, both Integer Linear Programming (ILP) and Satisfiability Modulo Theories (SMT) solving can be considered specialized forms of CP. However, unlike ILP solvers, CP solvers employ techniques such as backtracking, local search, and constraint propagation to explore the solution space. Additionally, a common characteristic of CP is the restriction of variable domains to finite sets. CP-SAT [63] is a widely utilized CP solver that exemplifies these principles.

2.5.4 Heuristic Approaches

Exact methods often struggle to solve TSN scheduling problems for large-scale networks within acceptable time limits, making heuristic algorithms a practical alternative. These methods provide near-optimal solutions without the computational demands of ensuring global optimality, which is essential for real-world TSN

applications requiring swift scheduling decisions. Various heuristic approaches have proven effective for TSN scheduling, each suited to specific scenarios and balancing trade-offs between solution quality and computational efficiency. By leveraging problem-specific insights or exploiting the structural properties of TSN networks, these methods guide the search for effective, albeit not always optimal, schedules. Examples include Genetic Algorithms (GA) [64], Simulated Annealing (SA) [65], Tabu Search (TS) [56], and Greedy Randomized Adaptive Search Procedure (GRASP) [66]. The choice of heuristic depends on factors such as network size, traffic patterns, and the desired balance between performance and computation time. Their effectiveness is generally assessed through simulations and comparisons with optimal or alternative heuristic solutions, using metrics like schedule feasibility, latency, jitter, and bandwidth utilization.

2.6 Joint Scheduling and Routing (JSR)

While TSN standards primarily address various aspects of achieving deterministic, low-latency communication, researchers have placed significant emphasis on scheduling Time-Triggered (TT) traffic, which represents critical data requiring timely and reliable transmission. Historically, scheduling and routing have been investigated as separate concerns within TSN. However, recent years have witnessed a rapid shift towards joint optimization approaches that offer significant advantages for enhancing TSN communication.

Routing and scheduling optimization exhibit a complementary relationship. In any network, and particularly within TSN, network topology and traffic behavior inherently influence the feasibility and optimality of schedules. Disregarding the

routing paths of traffic flows and neglecting to ensure they traverse the most efficient routes can negatively impact schedules and lead to resource wastage. Consequently, numerous TSN studies have addressed the joint optimization of scheduling and routing, employing diverse modeling and algorithmic approaches to achieve this goal.

The recognition of the interdependence between routing and scheduling within TSN has led to a surge in research on joint optimization approaches. Early work by Smirnov et al. [67] introduced a novel method for designing networks handling diverse traffic, using Pseudo-Boolean constraints for integrated message routing and scheduling. This surpassed traditional, separate optimization methods, demonstrating advantages through scalability tests and a multi-objective automotive case study. Schweissguth et al. [68] offered a concurrent ILP formulation for routing and scheduling in Time-Triggered Ethernet (TTE) networks, improving schedulability and latency compared to fixed-routing approaches.

Falk et al. extended an ILP model [46], presented in [56], for optimal TSN configuration, showing strong performance for moderately-sized networks, with data flow count impacting solution time. Subsequently [69], they proposed a faster, lower-memory "configuration-conflict graph" method for minimizing delays, outperforming traditional ILP approaches. Li et al. [70] presented a dynamic reconfiguration approach for TT networks using the JILP and SCA algorithms, achieving a 50-fold speed improvement with minimal success rate loss. They also introduced the "reconfigurable depth" metric. Meanwhile, [71] analyzed TSN scheduling optimizations, finding that input data optimizations were most beneficial, while solver-specific optimizations could negatively impact performance.

Addressing the need for low-latency communication in domains such as automotive control, Nie et al. [72] proposed a TSN scheduling method prioritizing overall timely message delivery in real-time applications, improving on-time scheduling despite potential minor individual message delays. The authors in [73] presents a Hybrid Genetic Algorithm (HGA) for joint scheduling and routing in TSN, improving upon the Nawaz, Enscore, Ham (NEH) Algorithm [74]. The HGA's optimized solution representation and novel schedule compression technique enhanced network bandwidth utilization by effectively balancing network load.

Wang et al. [75] developed an Improved Ant Colony Optimization (IACO) algorithm for TSN task scheduling, improving convergence speed and solution quality over traditional methods [76]. IACO minimized delays and congestion, meeting TSN timing requirements in simulation. Future work suggested exploring dynamic scheduling for real-world adaptability. Min et al.'s [77] reinforcement learning-based scheduler, Reinforcement Learning-based Routing (TSLR), improves TSN schedulability and link utilization over Joint Routing and Scheduling (JRaS), ILP-Red+Conf+Adv (IRCA), and Simulated Annealing-based dynamic routing with PSS (SA/PSS), balancing load distribution and path length. However, scalability and training requirements need further development. The authors in [78] proposes the Tabu-RG algorithm, a Tabu search-based approach for jointly optimizing routing and GCL scheduling in TSN. This improves schedulability for TT and AVB flows, reducing computation time by 75% compared to traditional methods while maintaining solution quality. A mixed scheduling method further enhances efficiency for differentiated service flows.

Wang et al. [79] propose a two-stage approach for joint routing and scheduling in TSN: a heuristic algorithm for routing optimization followed by a genetic algorithm for scheduling. This method improves schedulability and runtime compared to traditional approaches, demonstrating high effectiveness even in larger networks. The authors in [80] presents a joint routing and scheduling optimization for TSN, using a heuristic for routing and a genetic algorithm for scheduling to minimize the non-schedulability ratio. Simulation results show improved schedulability and runtime compared to traditional methods, even in larger networks. Lin et al. [81] propose a joint routing and scheduling algorithm for TSN, considering link load and delay in routing and using a hybrid scheduling method (basic and hyper periods) with tabu search optimization. Simulation results show improved scheduling completion time compared to a reference algorithm, especially under increased network traffic.

Sun et al. [82] present an improved grey wolf optimization (IGWO) algorithm for joint routing and scheduling in in-vehicle TSN, modeled as a job-shop scheduling problem. IGWO incorporates Lévy flight, historical learning, and tabu search to optimize a priority-based E2E delay function. A dynamic load-balanced routing strategy further minimizes interference and delay. Simulations demonstrate improved efficiency compared to other methods, particularly in large-scale networks, and analyze the impact of flow periods and topology on E2E delay. Yang et al. [83] propose online routing and scheduling (ONRS) algorithm for deterministic networks in dynamic industrial automation. Combining offline backbone selection and scalable routing with online reuse-based scheduling, ONRS significantly improves scheduling success rate and reduces computation time compared to existing methods. Zheng et al. [84] introduce the Multipath Mix-Flow Scheduling (MPMFS) algorithm for TSN with

Concurrent Multipath Transmission (CMT) in an SDN framework. MPMFS dynamically manages flow splitting, routing, and scheduling, improving link utilization, delay, and jitter over single-path scheduling and exhibiting good scalability.

Yang et al. [85] and Gong et al. [86] address TSN scheduling of chain flows (sequentially connected devices) using the TC-Flow model [85]. This model, implemented in a dedicated CF-TSN architecture, uses ILP-based offline and heuristic online scheduling algorithms to improve the number of schedulable flows compared to existing methods. Gong et al. [86]'s TSN Chained Flow Scheduling (TCFS) uses specialized constraints and an ILP-based offline scheduler with a tabu search-based heuristic for real-time adjustments. Experimental results show improved schedulability of chain flows compared to benchmarks. Xu et al. [87] propose the LSSR architecture for TSN in large-scale IIoT environments. Using machine learning and traffic analysis, LSSR intelligently groups network streams, iteratively optimizing scheduling and routing for improved schedulability and scalability with moderate computational overhead.

Research on JSR optimization in TSN continues to produce innovative solutions addressing diverse challenges. He et al. [88] propose DeepScheduler, a deep reinforcement learning approach for TSN message scheduling. Learning directly from traffic patterns, DeepScheduler adapts to various network setups, offering significant speed improvements and higher message delivery rates compared to traditional methods in both simulated and real-world environments. Syed et al. [89] present a Mixed-Integer Programming (MIP)-based algorithm for scheduling and routing static

traffic in TSN to optimize for dynamic traffic. Experiments across six vehicle network configurations showed improved load balancing and dynamic traffic handling capacity. Syed et al. [90] evaluated four heuristics for dynamic scheduling and routing in TSN-based IVNs using a vector bin packing model and centralized network controller. The Bottleneck heuristic showed the best performance, scheduling 16-22% more traffic streams than other methods.

The authors of [91] presents two multi-objective optimization methods (MILP and Simulated Annealing) for joint vPLC placement, scheduling, and routing in industrial networks to minimize latency and balance load. Simulation results show the SA approach is superior for efficiently finding optimal configurations. Huang et al. [92] propose the Incremental Routing and Scheduling (IRAS) algorithm to enhance TSN's TAS functionality. IRAS supports per-flow configuration, maximizing schedulable streams and bandwidth efficiency using variable time slots and optimized start times. Experimental results demonstrate near-optimal performance (96.5%) with feasible configuration times for large numbers of streams.

Mahfouzi et al. [93], [94] integrated control system stability analysis into the joint scheduling and routing optimization for cyber-physical control applications, using SMT to generate stable, real-time network configurations. New heuristics improved synthesis efficiency with minimal impact on solution quality, demonstrated effectively through experimental evaluation, including an automotive use case. Xu et al. [95] propose an integrated scheduling and routing solution for deterministic transmission in multi-hop, multi-queue TSNs using SMT and OMT. Optimizations including intelligent cycle offsetting and a "Listeners" based mapping approach improve

network utilization and schedulability for time-critical traffic compared to traditional methods.

The authors of [96] presents an SMT-based synthesis method integrating frame preemption (802.1Qbu) with traditional TSN scheduling to provide both predictability and flexibility. The approach determines message schedules, preemption points, and traffic allocation, incorporating routing constraints and preemption overheads for practical real-world deployments. Vlk et al. [97] developed two Constraint Programming (CP) models for joint scheduling and routing in TSN, one prioritizing optimal solutions and the other prioritizing speed. Using Logic-Based Benders Decomposition, these models significantly improved the network's ability to meet timing deadlines compared to traditional methods, highlighting the benefits of intelligent routing.

2.6.1 Multicast

Several researchers have incorporated specific stream characteristics, such as multicast transmission, into their JSR approaches for TSN. Multicast refers to the ability to transmit a stream from a single source end-system to multiple destination end-systems, delivering identical data to all recipients. Extending their prior work [68], Schweissguth et al. [98] presents an optimized ILP model for joint TSN routing and transmission scheduling, supporting multicast flows and fine-grained control. Model optimizations reduce size and solver runtime without compromising solution quality, enabling achievement of various network performance goals through different objective functions.

Li et al. [99] propose a novel ILP-based algorithm for optimizing multicast time-sensitive traffic in dynamic TSNs. Using network pruning and a "Clustered ILP" (CILP) approach to group flows based on timing and spatial properties, the algorithm accelerates schedule calculation while maintaining resource utilization efficiency. Pahlevan [100], [101] presents two heuristic JSR approaches supporting multicast: a genetic algorithm [100] integrating routing and scheduling for improved schedulability, transmission efficiency, and resource utilization; and the Heuristic List Scheduler (HLS) [101], enabling single-step schedule calculation for complex scenarios including multicast and inter-flow dependencies, significantly improving schedulability.

Yu et al. [102] propose the Minimal Distance Tree Construction - Heuristic Breadth First Search (MDTC-HB) framework for rapid network reconfiguration during VM migrations in multicast TSN. Using a two-phase approach with offline Minimal Distance Tree (MDT)-based pre-calculation and online heuristic-based adjustment, MDTC-HB minimizes recalculation, offering faster response times than existing approaches, validated on a 24-port TSN switch. The authors of [103] introduces the Adaptive Group Routing and Scheduling (AGRS) approach for optimizing multicast traffic in TSN. AGRS uses a three-phase strategy (pre-processing network simplification, schedule synthesis, post-processing refinement) to significantly improve schedulability (68% more flows) and computational efficiency (74% reduction in calculation time), validated on a 24-port TSN switch.

2.6.2 Task Scheduling

In TSN, tasks representing applications with periodic execution need rely on data from Time-Triggered (TT) streams and may generate TT streams themselves. Advanced scheduling algorithms address this by jointly scheduling tasks and TT streams, considering their interdependencies and timing constraints. This integrated approach optimizes network resource utilization, ensuring tasks receive necessary data on time and critical data is delivered reliably, even amidst disruptions. Constraint Programming and heuristic algorithms are examples of methods used for joint scheduling, enabling TSN to meet the demands of time-sensitive applications with greater efficiency and dependability.

Chahed et al. [104], present GenTSN, a hybrid genetic algorithm designed to optimize Time-Sensitive Networking (TSN) configurations in virtualized edge-compute environments. The algorithm addresses the joint problem of task placement, routing, and scheduling, utilizing a flexible genome encoding adaptable to diverse network configurations. Evaluation results demonstrate GenTSN's efficiency and scalability, achieving reduced makespan compared to fixed-route scheduling. GenTSN shows promise in enhancing TSN performance in virtualized environments, with future research directions including refining genetic operators and investigating online reconfiguration strategies.

2.6.3 Non-Critical Traffic Consideration

While TSN prioritizes scheduling Time-Triggered (TT) streams due to their critical nature, this can negatively impact non-critical traffic such as Audio Video Bridging (AVB) and Best-Effort (BE). Several researchers have addressed this concern by

considering non-critical traffic in their scheduling approaches. Gavriluţ et al. [40] present a joint TT routing and scheduling optimization approach that explicitly maintains AVB schedulability, integrating the K-Shortest Path (KSP) heuristic with a GRASP-based metaheuristic. Evaluations using synthetic and real-world test cases demonstrate effective coexistence of TT and AVB traffic.

Alnajim et al. [105] propose an incremental approach for routing and scheduling in TSN to maximize supported data flows, overcoming limitations of traditional methods. A path selection algorithm prioritizes bandwidth, hops, and load; new incremental scheduling algorithms (SWOTS-ASAP, SWOTS-ASAP-WS, SWOTS-AEAP-WS) maximize scheduled flows and minimize wasted bandwidth, demonstrating the effectiveness of incremental approaches with QoS data utilization. Li et al. [106] present a novel JSR optimization algorithm for TSN that proactively minimizes conflicts during route selection and uses "protected windows" to satisfy GCL constraints. Simulations and scalability tests demonstrate its ability to handle thousands of flows in under a second, outperforming ILP, DoC-aware, and HGA-based methods in mixed-traffic environments.

Gavriluţ et al. [107] present a TSN-aware scheduling heuristic and a Tabu Search-based metaheuristic (TTA) for optimizing traffic class assignment (TT, AVB, or BE) in distributed systems with HRT, SRT, and NC applications. TTA ensures HRT deadlines while maximizing SRT QoS, demonstrated effective through evaluation with realistic scenarios. Berisa et al. [108] present a new worst-case response time (WCRT) analysis for AVB traffic in TSN considering frame preemption, multiple queues, and CBS behavior. This analysis is integrated into a heuristic algorithm that

generates ST (or TT) network schedules guaranteeing AVB schedulability, using strategic frame preemption and a novel routing mechanism, demonstrated effective through comprehensive evaluations.

The authors in [77] proposes the Tabu-RG algorithm, a Tabu search-based approach for jointly optimizing routing and GCL scheduling in TSN to improve schedulability for TT and AVB flows. Tabu-RG achieves a 75% reduction in computation time compared to traditional solvers while maintaining performance, and a mixed scheduling method further enhances efficiency for differentiated service flows. Yang et al. [109] propose a joint scheduling and routing optimization for TSN combining GCNs and DRL. GCNs, using a simplified Chebyshev polynomial kernel for real-time performance, capture spatial dependencies, while DRL learns optimal decisions from experience using priority experience replay. Evaluations demonstrate good convergence and reduce average end-to-end delays compared to benchmarks.

Table 2.3 provides an overview of research studies focused on joint scheduling and routing. The table highlights key aspects of each study, including the solution approach (e.g., ILP, SMT, CP, Heuristic, etc.), the primary objective category (e.g., Topology Synthesis, Routing, Latency minimization, etc.), and the problem extensions addressed (Queuing, Multicast, Task Scheduling, and Non-Critical Traffic). This structured presentation allows for a clear comparison of the methodologies, objectives, and problem domains explored in the respective studies.

Table 2.3 : Summary of Solution Approaches, Objective Category and Problem Extensions of Joint Scheduling and Routing Approaches

Ref. /Year	Solution Approach	Objective Category	Problem Extensions			
			Queuing	Multicast	Task Scheduling	Non-Critical Traffic
[67] (2017)	PBO	Non-Critical Traffic	-	✓	-	BE
[46] (2018)	ILP	No Objective	-	-	-	-
[93] (2018)	SMT	No Objective	✓	-	✓	-
[100] (2018)	GA	Latency and Jitter	-	-	✓	BE
[40] (2018)	GRASP	Non-Critical Traffic	✓	-	-	AVB
[101] (2019)	LS, Heuristic	Latency and Jitter	-	-	✓	BE
[105] (2019)	Heuristic	No Objective	✓	-	-	BE
[73] (2020)	GA	Latency and Jitter	✓	-	-	BE
[69] (2020)	Heuristic	No Objective	-	-	-	-
[89] (2020)	ILP	Other	-	-	-	-
[102] (2020)	Heuristic	Routing	✓	✓	-	-
[103] (2020)	ILP	Routing	✓	✓	-	-
[95] (2020)	SMT	Topology Synthesis	✓	-	-	-
[98] (2020)	ILP	Latency and Jitter	-	✓	-	-
[75] (2021)	ACO	Routing	-	-	-	-

Table 2.3 : Continued

Ref. /Year	Solution Approach	Objective Category	Problem Extensions			
			Queuing	Multicast	Task Scheduling	Non-Critical Traffic
[97] (2021)	CP	Queuing	✓	-	-	-
[90] (2021)	Heuristic	No Objective	-	-	-	-
[92] (2021)	ILP	Latency and Jitter	-	-	-	-
[94] (2021)	SMT	No Objective	✓	-	✓	-
[99] (2021)	ILP	Routing	✓	✓	-	-
[72] (2022)	ILP	Latency and Jitter	-	-	-	-
[87] (2022)	SMT	No Objective	✓	-	-	-
[70] (2022)	ILP, Heuristic	Routing	-	-	-	-
[85] (2022)	Tabu Search	Other	✓	-	-	-
[91] (2022)	SA	Other	✓	-	-	-
[108] (2022)	Heuristic	Non-Critical Traffic	✓	-	-	AVB
[106] (2022)	Heuristic	No Objective	✓	-	-	BE
[109] (2022)	Machine Learning	Latency and Jitter	✓	-	-	BE, AVB
[88] (2023)	Machine Learning	No Objective	✓	-	-	-
[86] (2023)	Tabu Search	Other	✓	-	-	-

Table 2.3 : Continued

Ref. /Year	Solution Approach	Objective Category	Problem Extensions			
			Queuing	Multicast	Task Scheduling	Non-Critical Traffic
[77] (2023)	Machine Learning	Latency and Jitter	-	-	-	-
[78] (2023)	Tabu Search	Other	-	-	-	AVB
[79] (2023)	Heuristic, GA	No Objective	✓	-	-	-
[80] (2023)	GA	No Objective	✓	-	-	-
[81] (2023)	Tabu Search	No Objective	-	-	-	-
[82] (2024)	Heuristic	Latency and Jitter	✓	-	-	-
[83] (2024)	Heuristic	Other	✓	-	-	-
[104] (2024)	GA	Other	-	✓	✓	-

2.7 Reliability for TSN

Reliability is a fundamental concern in TSN, ensuring the delivery of transmitted streams without loss even in the face of obstacles such as device failures, network congestion, and transmission errors. In industrial, automotive, and aerospace applications where TSN is deployed, even momentary communication disruptions can lead to severe consequences, including system failures or safety hazards. To address these challenges, TSN implements multiple reliability mechanisms including redundancy, error correction, and dedicated path control and reservation for critical streams. The IEEE 802.1CB Frame Replication and Elimination for Reliability

(FRER) standard, for instance, provides seamless redundancy by duplicating critical frames across different network paths, while time-aware traffic shaping and precise scheduling works together to prevent congestion and maintain reliable communication channels for time-critical applications.

2.7.1 Frame Replication and Elimination

The IEEE 802.1CB standard introduces Frame Replication and Elimination for Reliability (FRER) as a key mechanism to enhance TSN reliability. FRER mitigates frame loss risks from link or switch failures by replicating frames of designated streams. These duplicates are sent along diverse paths, each carrying a unique sequence ID. At the destination, duplicates are eliminated, and missing frames are detected, ensuring dependable critical data delivery. This redundancy-based approach ensures data integrity by delivering the original information without loss, even if some paths are disrupted, as long as one replica reaches the destination [110]. By utilizing multiple transmission paths, FRER enhances TSN network resilience and ensures reliable delivery of critical data, even during network failures.

While the FRER mechanism, as defined in IEEE 802.1CB, effectively addresses frame replication and elimination processes, it does not explicitly delve into the intricacies of path selection for the replicas or their integration with TAS scheduling [111]. These aspects present significant optimization challenges that directly impact the overall effectiveness and efficiency of FRER implementation within TSN networks. Careful consideration of path diversity and coordination with TAS scheduling is crucial to ensure optimal performance and maximize the benefits of redundancy while minimizing potential overhead.

TSN's approach to reliability extends beyond FRER, encompassing a comprehensive set of tools and mechanisms that address various aspects of network resilience. FRER primarily focuses on packet-level redundancy, ensuring the survival of critical data even in the face of isolated link or switch failures. This complements traditional network redundancy protocols, such as Rapid Spanning Tree, by providing an additional layer of protection with minimal added delays [110], [112]. Additionally, TSN standards incorporate mechanisms for swift error detection and mitigation, allowing for the identification of network issues or misbehaving devices and triggering corrective actions to prevent widespread problems. Path control and bandwidth reservation further enhance reliability by reducing the likelihood of time-sensitive data encountering congestion-related delays or drops.

Finally, TSN includes standards for streamlined network configuration, minimizing the potential for human-caused errors that could compromise reliability [112]. This multifaceted approach to reliability solidifies TSN's position as a robust foundation for building dependable real-time communication networks, particularly for mission-critical applications where uninterrupted and accurate data transmission is essential.

2.7.2 Research on Reliability for TSN

Several research efforts have integrated reliability considerations into their joint scheduling and routing (JSR) approaches for TSN. Atallah et al. [113] present a greedy heuristic algorithm for designing fault-tolerant TSNs carrying TT traffic. Simultaneously optimizing topology, routing, and scheduling to ensure redundant paths for critical flows while minimizing costs, the algorithm achieves up to 50% cost reduction compared to traditional methods and scales effectively to large networks,

demonstrated via 380 synthetic test cases. Atallah et al. [47] propose the DoC-Aware Iterative Routing and Scheduling (DA/IRS) method, using Iterated ILP-based Scheduling (IIS), a Degree of Conflict (DoC) metric for stream partitioning, and DoC-Aware Multipath Routing (DAMR) to improve reliability and scalability, especially in high-utilization scenarios, as demonstrated by evaluation on 200 synthetic test cases.

The authors in [114] introduces a novel on-the-fly repair algorithm for maintaining strict timing requirements in TT networks during link failures. By strategically inserting empty time slots to enhance "schedule reparability" (a new ILP formulation), the algorithm achieves millisecond repair times and significantly improves recovery chances compared to full network recalculation. Syed et al. [115] analyzed four heuristics for dynamically scheduling and routing traffic across redundant paths using Frame Replication and Elimination for Reliability (FRER) in TSN-based in-vehicle networks. The Fault-Tolerant Bottleneck Heuristic (FTBH) performed best, maximizing schedulable traffic and minimizing reconfiguration time, using a vector bin packing model as in [90].

The authors of [116] presents a novel SMT-based method for calculating optimized TSN schedules and routes that meet both end-to-end deadlines and reliability targets. The method incorporates a fault model and redundant paths, using heuristics for real-world feasibility, and demonstrates effectiveness through experimental evaluation. Zhou et al. [117] integrated Automotive Safety Integrity Level (ASIL) decomposition (ISO 26262 [118]) with TSN message routing and scheduling to reduce system cost while maintaining safety by strategically using lower-ASIL components. An SMT-

based formulation and optimization algorithms, validated through experiments and a real-world case study, demonstrate significant cost savings.

Gavriluţ et al. [119] present three approaches for designing fault-tolerant TSN networks for safety-critical systems with zero downtime, using the Urgency-Based Scheduler (UBS): a fast heuristic, a GRASP metaheuristic, and a Constraint Programming model. These solutions automatically generate network layouts with backup routes for essential data, successfully tested including a real-world General Motors scenario. Li et al. [120] present heuristic algorithms for optimizing reliable TSN, combining path and seamless redundancy to maximize throughput and balance load while prioritizing timely critical data delivery. Their approach, including enhancements to the standard TSN scheduling mechanism, was validated on a custom testbed, demonstrating excellent throughput, load balancing, and reliable time-sensitive data delivery.

Min et al. [111] addresses the interplay of multipath routing and TAS scheduling within FRER in TSN. The proposed Member Wait-and-Forward (MWF) technique, with a deadlock-prevention solution, Redundancy-Weighted Multipath Routing (RWMR), and a Last Arrival Time-based TAS Scheduler (LATS) with metaheuristic optimization, improve concurrent flow support, reduce resource utilization, and enhance schedulability within the FRER framework. The authors in [121] introduces a FRER mechanism for improving data transmission in time-sensitive software-defined networking (TSSDN) for Industrial Internet of Things (IIoT) systems, balancing low latency and reliability. An end-to-end delay bound model and reliability

model assess FRER costs, while a heuristic algorithm maximizes system utility within resource constraints by dynamically computing routing and scheduling strategies.

The authors in [122] employ Constraint Programming (CP) to optimize distributed, safety-critical TSN systems. Their approach integrates static cyclic scheduling, redundant routing paths, and the Timed Efficient Stream Loss-Tolerant Authentication (TESLA) protocol for security. The CP model ensures adherence to stringent timing constraints, security protocols, and fault tolerance. The effectiveness of this integrated approach is demonstrated through comprehensive evaluation, including case studies potentially based on real-world deployments, showcasing its ability to deliver robust configurations for safety-critical TSN applications. Table 2.4 presents a summary of research studies centered on reliability in joint scheduling and routing. It outlines the key features of each study, such as the solution approach, primary objective category, and addressed problem extensions. This organized format facilitates a clear comparison of the methodologies, goals, and problem areas investigated in the studies.

Table 2.4 : Summary of Research on Reliable Joint Scheduling and Routing

Ref. /Year	Solution Approach	Objective Category	Problem Extensions			
			Queuing	Multicast	Task Scheduling	Non-Critical Traffic
[119] (2017)	GRASP, Heuristic	Topology Synthesis	-	✓	-	-
[113] (2018)	Heuristic	No Objective	-	-	-	-
[114] (2018)	ILP	Reliability	-	✓	-	-
[47] (2020)	ILP	No Objective	✓	-	-	-
[122] (2020)	CP	Topology Synthesis	-	-	✓	-
[115] (2021)	Heuristic	No Objective	-	-	-	-
[116] (2021)	SMT	No Objective	-	-	-	-
[117] (2021)	SMT	Topology Synthesis	-	-	-	-
[120] (2021)	Heuristic	Routing	✓	✓	-	-
[57] (2022)	CP, SA, LS	Topology Synthesis	✓	✓	✓	-
[111] (2023)	Heuristic	Reliability	✓	-	-	-
[121] (2024)	Heuristic	Reliability	✓	-	-	-

The benchmark work in this thesis by Reusch et al. [57] introduces a novel approach for ensuring reliability and timing guarantees in TSN. Their method utilizes CP to formulate the routing and scheduling problem for TT streams, considering strict timing, bandwidth, and dependability constraints. Crucially, the CP model incorporates reliability by including redundant paths and backup schedules, allowing

the system to tolerate link or node failures without compromising real-time performance. This approach enables efficient exploration of the solution space, leading to optimal or near-optimal configurations for moderately sized networks (up to 32 end systems and 16 bridges), significantly improving upon traditional methods. While the paper also explores Simulated Annealing for larger networks, the focus remains on CP's ability to provide provably feasible and optimal solutions.

Experimental results validate the effectiveness of the CP-based method, demonstrating superior performance compared to baseline approaches in terms of solution quality and scalability. This work effectively showcases CP's strength as a powerful and flexible tool for addressing complex multi-objective problems in real-time networking, particularly when dependability and timing constraints are paramount.

2.8 Problem Instances Characteristics

To validate problem instances, a review of existing instances is essential. The number of end-systems and bridges varies across prior works. In CP based studies, the number of end-systems ranges from 2 to 128, while the number of bridges ranges from 2 to 64 [5]. The benchmark study by Reusch et al. [57] generated problem instances with up to 128 end-systems and 64 bridges, although their CP model successfully solved instances with only 32 end-systems and 16 bridges. This indicates that the creation of large instances does not guarantee solvability by the model.

Furthermore, the number of transmission streams (TT streams) in CP-based problem instances ranges from 2 to 300 [5]. Analogously to the number of nodes, a large

number of streams does not inherently imply the model's capacity to solve the instance. Overall, as reviewed by Stuber et al. [5], problem instances can range from 2 end-systems, 2 bridges, and 2 streams to arbitrarily large values; however, simply creating a large instance does not ensure a solution can be found. More realistic problem instance creation would involve drawing upon and mirroring instances previously created in prior work.

2.9 Performance Evaluation Metrics

In TSN, the effectiveness of scheduling and routing algorithms is quantified through several critical performance metrics. These metrics serve as benchmarks to evaluate the quality of solutions and their practical applicability in real-world networks. This section details the metrics used in this thesis to assess TSN implementations: response time, latency, worst-case latency, and scalability. These metrics comprehensively assess the effectiveness and practicality of TSN scheduling and routing solutions.

2.9.1 Response Time

In TSN, response time is the total duration a model, scheduling algorithm, or routing algorithm requires to generate a valid solution. This metric is crucial for assessing the model's ability to find a feasible or optimal solution—the best path, schedule, or both concurrently. A shorter response time directly improves the network's ability to adapt quickly to dynamic traffic demands and maintain the deterministic communication essential for TSN's real-time applications. JSR research showed response times ranging from 14ms for small networks to a maximum of 172,800,000ms for the largest networks. CP-based approaches within this research showed a similar upper bound, with minimum response times of 100,000ms [5]. Factors influencing response time

include algorithm complexity, network size and topology, and available computational resources.

In TSN research, this response time is often referred to interchangeably as optimization time or runtime [5], [57]. Minimizing response time is paramount for applications demanding low latency, such as industrial automation, automotive control systems, and medical devices, where delays can have significant consequences. Therefore, ongoing research focuses on developing algorithms and architectures that can reduce response time while maintaining or improving solution quality.

2.9.2 Latency

Latency measures the time taken for data transmissions, such as streams, frames, tasks or packets, to travel from source to destination within a network. In TSN applications, particularly in demanding environments like industrial automation and automotive systems, average latency requirements are significantly more stringent than traditional best-effort networks. Latency is a fundamental metric for evaluating whether a scheduling solution can meet the time-critical requirements of TT Traffic. Successful TSN implementations must maintain latency values within these bounds to ensure reliable real-time communication.

Effective TSN implementations must ensure predictable, consistently low latency and minimal jitter to support reliable, deterministic real-time communication. Exceeding latency thresholds can result in malfunctions, data loss, or failures in safety-critical systems. Key factors affecting latency include transmission, processing, queuing, and propagation delays. Optimizing these factors is essential to meet the stringent

performance demands of TSN applications, with specific latency requirements dictated by the application's timing constraints and the consequences of exceeding them.

2.9.3 Worst-Case Latency

Worst-case latency represents the maximum possible delay that any stream, frame, task, or packet might experience within the network. This metric is particularly critical for safety-critical applications demanding predictable behavior. The ability to bound and guarantee worst-case latency is a key differentiator of TSN compared to traditional Ethernet networks, as it ensures that time-sensitive communications will always meet their deadlines, even under peak load or adverse conditions. The ability to bound and guarantee worst-case latency is a key differentiator of TSN compared to traditional Ethernet networks.

2.9.4 Scalability

Scalability measures a scheduling solution's performance as network size and complexity increase. This metric is evaluated across several dimensions: network size (number of nodes and links), traffic volume (number of streams), and computational resource requirements. Practical TSN implementations must maintain acceptable performance when scaling to larger networks of nodes and streams. Within the JSR research, CP-based approaches demonstrated scalability across networks ranging from 20 to 96 nodes (end-systems and bridges) [5]. A solution's scalability directly impacts its viability for large-scale industrial deployments.

2.10 Research Gap

The literature review reveals considerable advancements in TSN scheduling and routing, especially for TT traffic. However, significant gaps remain. Existing Joint Scheduling and Routing (JSR) approaches for TT traffic primarily focus on specific optimization algorithms, such as ILP, heuristics, and reinforcement learning, without addressing latency minimization, scalability, and diverse traffic management within a unified framework. These methods often fall short in large-scale networks with complex topologies and numerous high-priority flows, resulting in suboptimal latency and schedulability under high load. Furthermore, the absence of a standardized bridge management approach limits the effectiveness of current JSR solutions, which struggle to handle TT and BE traffic simultaneously within a single, scalable, and low-latency framework.

Additionally, current research insufficiently addresses BE traffic scheduling and reliability mechanisms in TSN. Most studies emphasize TT traffic, neglecting the need for balanced scheduling that considers both TT and BE traffic. This oversight leads to reduced schedulability and performance for BE traffic, which is important in TSN deployments. Furthermore, Existing TSN redundancy mechanisms are either insufficient or overly complex. A unified JSR framework integrating temporal and spatial redundancy is needed to address reliable TT communication under different scenarios, a challenge yet to be addressed effectively in prior research.

2.11 Chapter Summary

This chapter presents a comprehensive literature review on TSN. It begins with an introduction and background information on TSN, including its architecture. The

chapter then delves into TSN standards and mechanisms, with a focus on reliability. TSN traffic classes are discussed, followed by an in-depth exploration of traffic shaping techniques, including TAS, CBS, and ATS. A significant portion of the chapter is dedicated to scheduling in TSN. It covers the scheduling problem, its complexity, and various scheduling approaches. The chapter then expands on the concept of Joint Scheduling and Routing (JSR), addressing multicast scenarios, task scheduling, reliability considerations, and the handling of non-critical traffic. A key research gap lies in the lack of a unified JSR framework for TSN that simultaneously optimizes latency for TT traffic while ensuring schedulability and performance for BE traffic in large-scale networks. Existing approaches often neglect BE traffic scheduling, lack scalability, and fail to address bridge management and reliability concerns effectively.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This chapter explores scheduling and routing techniques for reliable data transmission in Time-Sensitive Networking (TSN), with a focus on redundancy mechanisms. The system model underpinning the proposed approaches is presented, followed by a detailed description of the network bridge design and Queue-Gating Time mechanism. An exemplary problem instance illustrates the model design in a practical context. The chapter then explores the optimization framework, outlining algorithms and methodologies for achieving optimal scheduling and routing solutions. Specific constraints to ensure feasibility and practicality are discussed, followed by objective functions that guide the optimization process.

Figure 3.1 provides a visual representation of a high-level process flow for an optimization framework designed for joint scheduling and routing in TSN. The framework begins with inputs from the System Model, which includes the Network Model, Application Model, and Queue-Gating Time (QGT) component. The framework is divided into two parallel paths: Routing and Scheduling. Both paths incorporate the OHDSR model, which includes scheduling and routing aspects with constraints such as Priority Scheduling Constraints, Stream Scheduling Constraints, Task Scheduling Constraints, and Routing Constraints to optimize network performance. The OHDSR+ model extends OHDSR by including Stream Path Reliability Constraints in the routing section and Stream Schedule Reliability Constraints in the scheduling section, enhancing reliability in data transmission.

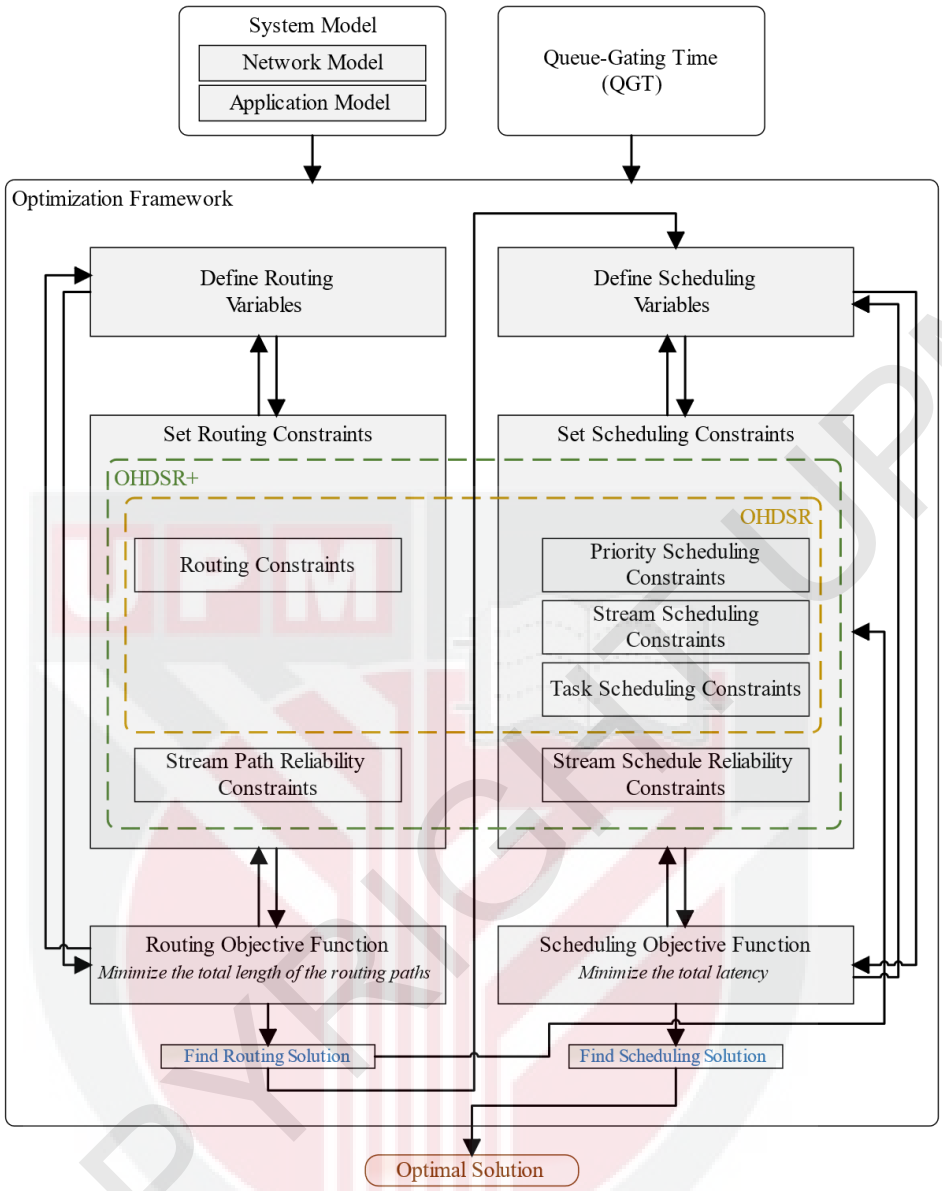

Figure 3.1 : Process Flow of Our Optimization Framework

In the Routing path, the objective function aims to minimize the total length of routing paths, leading to the Find Routing Solution stage. Conversely, the Scheduling path's objective function focuses on minimizing the total latency of routing paths, culminating in the Find Scheduling Solution stage. Finally, the outcomes from both paths are combined to achieve an Optimal Solution that meets the joint scheduling and routing objectives, ensuring efficient and reliable traffic management in TSN.

3.2 System Model

This section details the mathematical foundation and conceptual framework of the time-sensitive network. A detailed environment is established to demonstrate network challenges and their solutions, beginning with the fundamental network model and communication principles, followed by the practical implementation of the application model.

3.2.1 Network Model

Imagine a network as a map, where devices are locations and connections are roads. This map, however, is more than just lines and dots – it's a dynamic system represented by a directed graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$. Each device, whether it's a computer, server, or smartphone, is a node on this map, denoted as $v_a \in \mathcal{V}$, and the connections between them are edges, denoted as $(v_a,\ v_b) \in \mathcal{E}$, signifying the flow of data.

Within the set of nodes $\mathcal{V}$, there are two types: bridges (BR), acting as traffic directors like network switches, and end-systems (ES), the actual destinations and origins of data like our computers. Every connection between these nodes is full-duplex, allowing data to flow simultaneously in both directions.

When communication occurs, one end-system takes the role of the "talker," initiating the conversation by sending data packets, while the other becomes the "listener," receiving these packets. This simple model provides a foundational understanding of how networks function, serving as a starting point to explore more complex network structures and analyze their performance.

The network model depends on the precise synchronization provided by TSN, using the IEEE 802.1Qbv standard. This ensures coordinated operation between all network components, from bridges to end systems, allowing for accurate traffic management. End systems manage their tasks using internal schedulers, guaranteeing predictable and efficient execution within the overall synchronized network.

3.2.2 Application Model

Our application model, $\delta_k \in \Delta$, utilizes a directed acyclic graph (DAG) as its core representation. This DAG comprises a set of nodes with each node symbolizing a specific task, denoted as τ_j, within the application. Connecting these nodes is a set of edges, $\mathcal{E}$, which signify the flow of data communication between the tasks. Essentially, the edges illustrate how the output of one task serves as the input for another, creating a dependency chain within the application. Each application operates within a period (Γ_k^{δ}). The system hyperperiod (HP), calculated as the least common multiple (LCM) of all application periods ($HP = \text{lcm}(\{\Gamma_k^{\delta} | \delta_k \in \Delta\})$, provides a common timeframe for the coordinated execution of multiple applications.

Within our model, each application task ($\tau_j \in T$) is characterized by a tuple: (es_j^{τ}, ω_j^{τ}, Γ_j^{τ}), specifying the end-system (es_j^{τ}) of execution, (ω_j^{τ}) represents its worst-case execution time (WCET), and its period (Γ_j^{τ}). End-systems are classified as either 'talkers,' initiating data streams, or 'listeners,' receiving them. Communication within an end-system relies on mechanisms such as message queues or shared memory. However, inter-system communication, crucial for tasks distributed across multiple end-systems, is represented by data streams managed using Time-Sensitive

Networking (TSN). These streams, representing communication requirements, can be unicast (one listener) or multicast (multiple listeners).

The analysis focuses on multicast streams ($\sigma_i \in S$), each originating from a source task ($\tau_{talker}^{\sigma_i}$) and destined for multiple destination tasks ($T_{listner}^{\sigma_i}$). Each stream (σ_i) requires a dedicated path to ensure timely delivery (within its period, Γ_i^{σ}) to all destinations. The Maximum Transmission Unit (MTU), defining the maximum packet size, significantly impacts routing and scheduling decisions for multicast streams. The MTU imposes constraints on stream transmission. Streams exceeding the MTU must be fragmented, requiring careful scheduling to ensure timely delivery and avoid performance degradation or data loss.

3.3 Queue-Gating Time (QGT) for the Bridge

A standardized bridge architecture ($br_n \in BR$) is employed throughout the network to simplify management. Each bridge possesses eight queues ($q_\phi^{br_n} \in Q^{br_n}$), each assigned a data type (ρ_ϕ^q). For clarity, the bridge identifier is omitted as all bridges share an identical design. Streams (σ_i) are dynamically assigned to queues with matching data types based on queue storage levels to optimize resource utilization and prevent congestion. For example, let us consider queues q_0 and q_1 handle TT streams. Load balancing directs incoming TT streams to the queue with fewer active streams. In the case of equal load, the queue with lower index (q_1) is selected, ensuring predictable and fair queue allocation.

Each queue is associated with a dedicated gate ($g_q \in G_q$) regulating its outgoing data flow. Gate operation, determined by predefined Gate Control Lists (GCLs), dictates time-slotted transmission. Initially, all gates are closed to prevent unauthorized data transmission. Each GCL operates within a gate control period ($t_\varphi \in \Phi$), defining the state (ψ_g) of each associated gate for each queue (q_ϕ). An eight-digit binary number within the GCL specifies the gate states, with "1" representing open and "0" representing closed. The least significant bit corresponds to queue (q_0), and the most significant bit to queue (q_7), enabling fine-grained control and prioritized traffic scheduling.

The network employs two queue types: Time-Triggered (TT) and Best-Effort (BE). TT queues q_0 and q_1 manage time-critical streams requiring guaranteed delivery, while BE queues q_6 and q_7 handle less time-sensitive streams. Queues q_2 through q_5 are unused (N/U). The consistent queue configuration produces predictable gate state transitions within each gate control period. Gating cycles follow this pattern:

- At t_1: TT queues (q_0 and q_1) open ("00000011").
- At t_2: All gates closed ("00000000").
- At t_3: BE queues (q_6 and q_7) open ("11000000").
- At t_4: All gates closed again ("00000000"), completing the cycle.

Gate transition timing, as defined by the GCLs, is synchronized with the system's Hyperperiod (the least common multiple of all periodic task periods). This alignment, illustrated in Figure 3.2, optimizes network scheduling, ensuring timely resource

allocation for all tasks. This integrated approach to gate control, queue management, and timing demonstrates the sophisticated design of the network infrastructure.

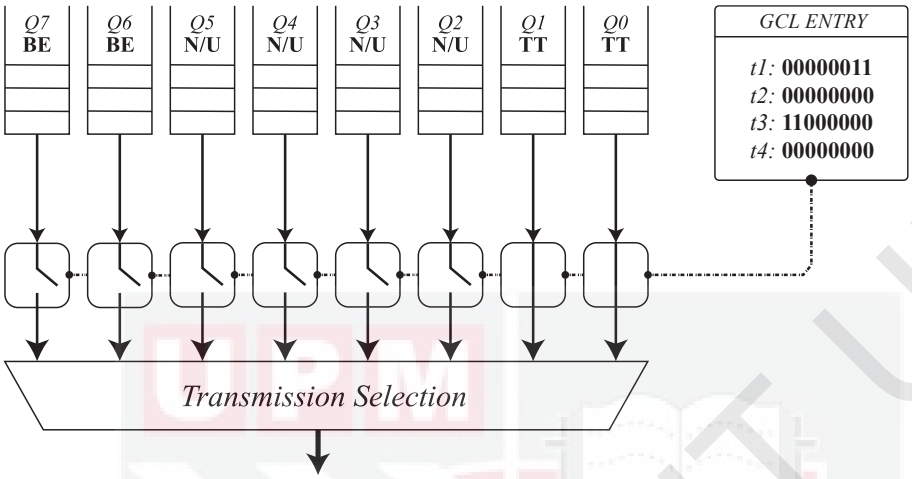

Figure 3.2 : A Simplified TSN Bridge Features Queues and Gates, Time-Triggered (TT) Traffic is Allocated to Queue 0 and 1, While Best Effort (BE) Traffic is in Queue 6 and 7. Queues 2-5 Are Unused (N/U)

3.3.1 The Modeling of QGT

Network behavior is analyzed using Queue-Gating Time (QGT, $\kappa_{\varphi}^{q}\boxed{\psi}$), a fine-grained model based on bridge gating times ($t_{\varphi} \in \Phi$) representing individual queue states throughout each gating cycle. At $t_1 = 0\mu s$ (GCL: 00000011), queues q_0 and q_1 are open ($\psi_0 = \psi_1 = 1$; $\kappa_1^0\boxed{1} = \kappa_1^1\boxed{1} = 0\mu s$), while queues q_5 through q_7 are closed (e.g., $\psi_5 = 0$; $\kappa_1^5\boxed{0} = 0\ \mu s$). The QGT model enables precise analysis of queue open times within the gating cycle, facilitating assessment of queue utilization, bottleneck identification, and data flow optimization.

Table 3.1 : A Queue-Gating Time (QGT) Determines When Each Queue is Open (Indicated by a Status of "1") or Closed (Indicated by a Status of "0") at a Particular Gating Time

$\kappa_{\varphi}^{q}\boxed{\psi}$	t_1	t_2	t_3	t_4
q_0	$\kappa_1^0\boxed{1}$	$\kappa_2^0\boxed{0}$	$\kappa_3^0\boxed{0}$	$\kappa_4^0\boxed{0}$
q_1	$\kappa_1^1\boxed{1}$	$\kappa_2^1\boxed{0}$	$\kappa_3^1\boxed{0}$	$\kappa_4^1\boxed{0}$
q_2	$\kappa_1^2\boxed{0}$	$\kappa_2^2\boxed{0}$	$\kappa_3^2\boxed{0}$	$\kappa_4^2\boxed{0}$
q_3	$\kappa_1^3\boxed{0}$	$\kappa_2^3\boxed{0}$	$\kappa_3^3\boxed{0}$	$\kappa_4^3\boxed{0}$
q_4	$\kappa_1^4\boxed{0}$	$\kappa_2^4\boxed{0}$	$\kappa_3^4\boxed{0}$	$\kappa_4^4\boxed{0}$
q_5	$\kappa_1^5\boxed{0}$	$\kappa_2^5\boxed{0}$	$\kappa_3^5\boxed{0}$	$\kappa_4^5\boxed{0}$
q_6	$\kappa_1^6\boxed{0}$	$\kappa_2^6\boxed{0}$	$\kappa_3^6\boxed{1}$	$\kappa_4^6\boxed{0}$
q_7	$\kappa_1^7\boxed{0}$	$\kappa_2^7\boxed{0}$	$\kappa_3^7\boxed{1}$	$\kappa_4^7\boxed{0}$

Table 3.1 presents QGT values ($\kappa_{\varphi}^{q}\boxed{\psi}$) in a matrix format, with rows representing queues (Q) and columns representing gating times (Φ). Each cell indicates a specific queue's QGT at a given gating time. The matrix is straightforward to interpret: each cell ($\kappa_{\varphi}^{q}\boxed{\psi}$) indicates whether a queue is open (1) or closed (0) during a specific gating time (t_{φ}). For example, $\kappa_1^1\boxed{1} = t_{\varphi}$ shows that queue q_1 is open at the first gating time, while $\kappa_3^3\boxed{\psi}$ reflects the gate status of q_1 at the third gating time.

3.4 Exemplary Problem Instances

This section analyzes network routing and scheduling challenges, considering data stream characteristics and network limitations. Two scenarios are examined: (1) basic routing and scheduling for unicast and multicast transmissions, focusing on optimal path selection while respecting bandwidth constraints and minimizing delays; and (2) routing and scheduling with redundancy to enhance reliability, which introduces complexities in managing duplicate data flows, requiring optimization of redundancy levels, path selection, and congestion control. Analysis of these scenarios informs the development of advanced network optimization algorithms.

3.4.1 Problem Formulation

This research addresses scheduling and routing challenges in TSN systems, where interconnected bridges connect end-systems that generate data streams upon task completion. These streams flow from talker (source) to listener (destination) end-systems. Efficiently managing data stream flow requires both scheduling (optimizing transmission timing and order to meet deadlines and avoid conflicts) and routing (selecting optimal paths considering congestion, bandwidth, and latency).

This problem's complexity stems from varying stream characteristics (size, deadlines, redundancy), network topology, and bridge capabilities. This research aims to develop algorithms and strategies for seamless and reliable TSN communication under stringent, time-sensitive requirements. Optimal TSN performance requires synchronized execution of tasks and coordinated data stream transmission ("applications") to minimize latency and ensure responsiveness. This necessitates a multi-faceted approach.

This requires precise end-system task scheduling to avoid delays and resource conflicts, optimal stream routing across bridges considering bandwidth, congestion, and timing constraints, and robust network integrity to prevent data loss. Successful TSN systems require a balance between low latency (for responsiveness) and high reliability (for accurate, timely data delivery). This research develops algorithms and protocols to achieve this balance, enabling high performance in demanding industrial and automotive applications.

3.4.2 Unicast and Multicast Exemplary

To gain a comprehensive understanding of the network structure and its components, let's analyze Figure 3.3. The figure illustrates a network consisting of interconnected nodes ($\mathcal{V}$) and links ($\mathcal{E}$). Nodes represented as $\mathcal{V} = \{es_1, es_2, es_3, es_4, br_1, br_2, br_3, br_4\}$ serve as the fundamental units within the network. Connecting these nodes are links, denoted as $\mathcal{E} = \{(es_1, br_1), (es_1, br_2), (es_2, br_1), (es_2, br_2), (es_3, br_3), (es_3, br_4), (es_4, br_3), (es_4, br_4), (br_1, br_2), (br_1, br_3), (br_1, br_4), (br_1, es_1), (br_1, es_2), (br_2, br_1), (br_2, br_3), (br_2, br_4), (br_2, es_1), (br_2, es_2), (br_3, br_1), (br_3, br_2), (br_3, br_4), (br_3, es_3), (br_3, es_4), (br_4, br_1), (br_4, br_2), (br_4, br_3), (br_4, es_3), (br_4, es_4)\}$ which establish pathways for interaction and communication.

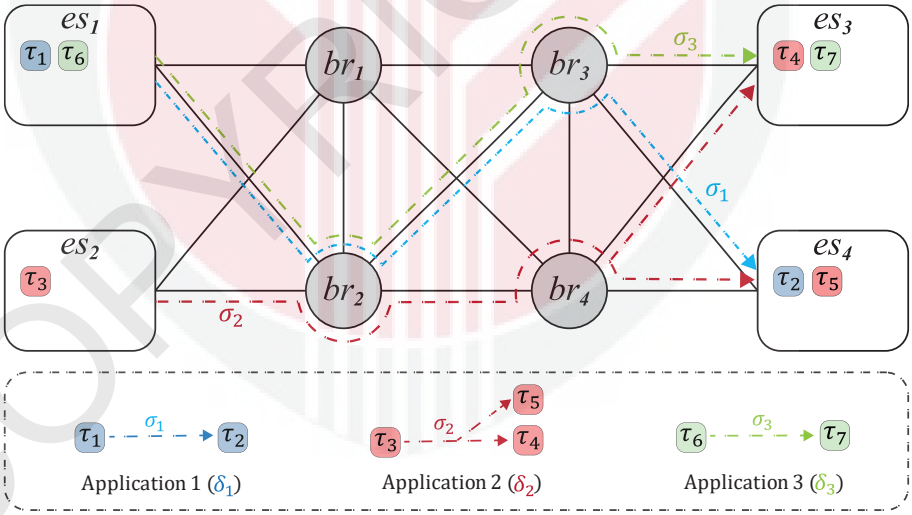

Figure 3.3 : The Network Topology of the Exemplary Problem Instance Illustrates the Route Path Taken by Each Stream without Considering Redundancy

Building upon the network structure (Figure 3.3), each link $(v_a, v_b) \in \mathcal{E}$ is assigned a speed $\Omega_{(v_a,v_b)} = 10$ Mbps, representing the data transfer rate. Propagation delays are

ignored for simplicity. Each end-system either transmits or receives data streams, managed within a single application encompassing associated actions. Application functionality may be independent or involve interactions with other tasks and data streams, depending on complexity.

The set $\Delta = \delta_1, \delta_2, \delta_3$ represents the applications within the network, each associated with a unique data stream ($\sigma_1, \sigma_2, \sigma_3$). Specifically, δ_1 and δ_3 comprise two tasks each, while δ_2 comprises three tasks. These application tasks, distributed across sender and receiver end-systems, communicate via dedicated streams, creating direct task-to-task communication channels for data exchange and coordinated processing. Data streams within the network exhibit varying sizes, with $L_1^\sigma = 70Bytes$, $L_2^\sigma = 80Bytes$, and $L_3^\sigma = 35Bytes$ for streams δ_1, δ_2 and δ_3, respectively. Furthermore, the worst-case execution times (WCET) ω_j^τ of tasks differ across applications: $30\mu s$ for δ_1 and δ_2, and $20\mu s$ for δ_3. These heterogeneities in stream sizes and WCETs pose significant challenges for efficient scheduling and resource allocation within the network. Table 3.2 summarizes the application, stream, and task parameters of the unicast and multicast exemplary problem instance, inspired by the motivational examples introduced in [57], [97].

Table 3.2 : The Applications, Streams, and Task Parameters of the Unicast and Multicast Exemplary Problem Instance

Δ	Γ_k^δ	T	ω_j^τ	Γ_j^τ	S	ρ_i^σ	L_i^σ	Γ_i^σ
δ_1	1000	τ_1, τ_2	30	1000	σ_1	TT	70	1000
δ_2	1000	τ_3, τ_4, τ_5	30	1000	σ_2	TT	80	1000
δ_3	1000	τ_6, τ_7	20	1000	σ_3	BE	35	1000

Network traffic is categorized into two priority levels. Applications δ_1 and δ_2 utilize a Time-Triggered (TT) protocol, ensuring deterministic transmission at predetermined intervals for control signals and sensor data typical of real-time systems (e.g., engine management, brake control, or industrial automation control messages). Conversely, application δ_3 operates on a Best-Effort (BE) basis, opportunistically using available resources without guaranteed timing or priority for general data transfers (e.g., file transfers, email, or system logs). For simplicity, all applications share a period and deadline of $1000\mu s$, resulting in a Hyperperiod (HP) equal to $1000\mu s$. This standardized period facilitates scheduling and analysis, demonstrating differentiated traffic management and prioritized handling of time-sensitive data.

Each of the four network bridges employs eight queues to manage data flow. Queues q_0 and q_1 are dedicated to Time-Triggered (TT) traffic, prioritizing time-sensitive data. Queues q_6 and q_7 handle Best-Effort (BE) traffic. The middle queues (q_2, q_3, q_4, and q_5) are currently unused, allowing for future expansion or additional traffic classes. This configuration emphasizes the distinct prioritization of TT and BE traffic.

A gating mechanism, with gating times (t_φ) culminating in $1000\mu s$, synchronizes with the network period and deadline. Gates open sequentially at 0, 400, 700, and $1000\mu s$. Time-Triggered (TT) gates (g_0, g_1) open at $0\mu s$ ($\psi_0 = \psi_1 = 1$), allowing transmission from queues q_0 and q_1, and close at $400\mu s$ ($\psi_0 = \psi_1 = 0$). This $400\mu s$ window prioritizes TT traffic. Conversely, Best-Effort (BE) gates (g_6 and g_7) remain closed ($\psi_6 = \psi_7 = 0$) until $700\mu s$, ensuring that TT traffic is transmitted uninterrupted. At this point, BE traffic transmission commences. Table 3.3 details these gate operations.

The remaining $300\mu s$ of the period ($1000\mu s$) is dedicated to BE traffic processing. This structured gating prioritizes TT traffic while enabling BE data transmission within available network capacity. The Gating Control List (GCL, Table 3.3) defines gate behavior. BE gates (g_6 and g_7) remain closed ($\psi_6 = \psi_7 = 0$) for the initial 700 µs, ensuring uninterrupted TT transmission. At $700\mu s$, these gates open ($\psi_6 = \psi_7 = 1$) for $300\mu s$, enabling BE traffic transmission before the period ends. Figure 3.4 visually illustrates the gate and queue configuration, demonstrating time allocations for each traffic type.

Table 3.3 : The Gate Control List (GCL) Entry Based on Different Gating Times

t_φ	GCL entry
$0\mu s$	00000011
$400\mu s$	00000000
$700\mu s$	11000000
$1000\mu s$	00000000

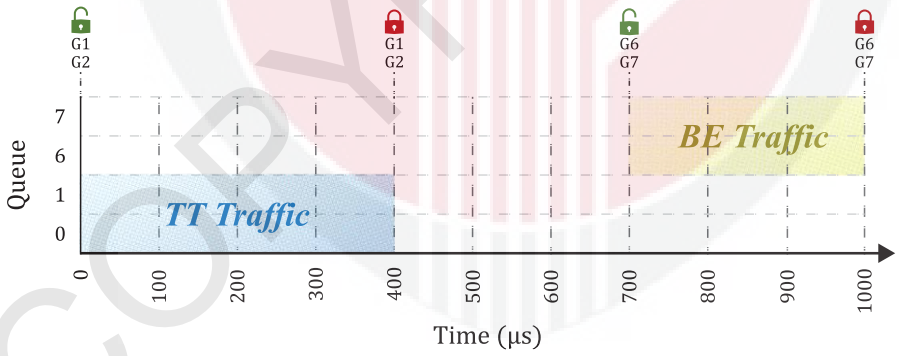

Figure 3.4 : The Opening and Closing Times of the Gates for Each Queue, along with the Type of Traffic Permitted for Transmission during Those Open Intervals

Table 3.4 presents the Queue Gating Table (QGT), derived from the Gating Control List (Table 3.3) and queue gating calculations (Section 3.3), providing a

comprehensive roadmap of queue states (open/closed) relative to gating times ($\kappa_{\varphi}^{q}[\psi]$). This detailed coordination ensures prioritized and efficient data flow.

Table 3.4 : The Values of the Queue-Gating Time (QGT) for Each Queue at a Particular Gating Time

$\kappa_{\varphi}^{q}[\psi]$	t_1= 0µs	t_2 = 400µs	t_3 = 700µs	t_4 = 1000µs
q_0	$\kappa_1^0[1]$= 0µs	$\kappa_2^0[0]$= 400µs	$\kappa_3^0[0]$= 700µs	$\kappa_4^0[0]$= 1000µs
q_1	$\kappa_1^1[1]$= 0µs	$\kappa_2^1[0]$= 400µs	$\kappa_3^1[0]$= 700µs	$\kappa_4^1[0]$= 1000µs
q_2	$\kappa_1^2[0]$= 0µs	$\kappa_2^2[0]$= 400µs	$\kappa_3^2[0]$= 700µs	$\kappa_4^2[0]$= 1000µs
q_3	$\kappa_1^3[0]$= 0µs	$\kappa_2^3[0]$= 400µs	$\kappa_3^3[0]$= 700µs	$\kappa_4^3[0]$= 1000µs
q_4	$\kappa_1^4[0]$= 0µs	$\kappa_2^4[0]$= 400µs	$\kappa_3^4[0]$= 700µs	$\kappa_4^4[0]$= 1000µs
q_5	$\kappa_1^5[0]$= 0µs	$\kappa_2^5[0]$= 400µs	$\kappa_3^5[0]$= 700µs	$\kappa_4^5[0]$= 1000µs
q_6	$\kappa_1^6[0]$= 0µs	$\kappa_2^6[0]$= 400µs	$\kappa_3^6[1]$= 700µs	$\kappa_4^6[0]$= 1000µs
q_7	$\kappa_1^7[0]$= 0µs	$\kappa_2^7[0]$= 400µs	$\kappa_3^7[1]$= 700µs	$\kappa_4^7[0]$= 1000µs

Figure 3.5 presents a Gantt chart visualizing task and stream scheduling within the network (Hyperperiod (HP) = $1000\mu s$). Distinct colors represent applications, highlighting the association between tasks and streams. The chart displays data frame transmissions between devices (end-systems and bridges), and the bottom four rows depict end-system tasks. The visualization clarifies the interplay of tasks, streams, and resources, and the effectiveness of traffic management. Stream σ_1 (light blue), originating at es_1 and transmitted to br_2, begins at $30\mu s$ (following task τ_1 completion) and ends at $86\mu s$. Transmission duration, calculated as stream size ($70Bytes$) divided by link speed ($10Mbps$), is $56\mu s$. Other stream durations are determined similarly, accounting for frame overhead. Half-tone blue and yellow rectangles denote TT and BE gate openings, respectively, on the relevant bridges, providing precise timing information. This layered visualization clarifies gate activity, bridge involvement, and their impact on network traffic flow.

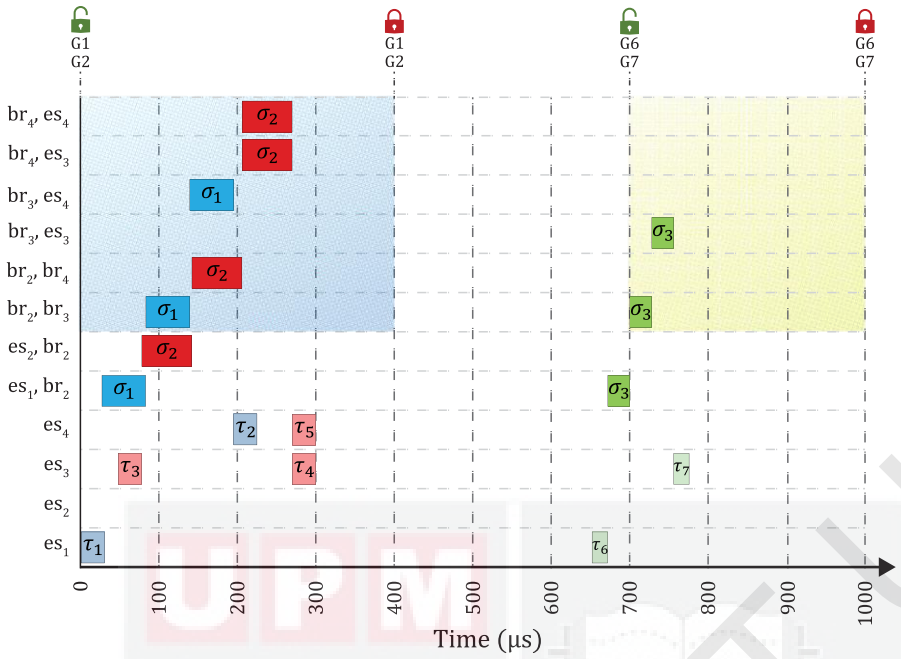

Figure 3.5 : The Network Schedule of the Exemplary Problem Instance for the Tasks and Streams

3.4.3 Considering Redundancy Exemplary

This section provides a practical example illustrating the system model, taking into account the redundancy of TT streams. Figure 3.6 depicts a 4-end-system, 4-bridge network topology. Each link operates at $10Mbps$. Three applications ($\Delta = \{\delta_1, \delta_2, \delta_3\}$) with varying task counts (δ_1, δ_2: 2 tasks, δ_3: 3 tasks) are supported. Streams ($\sigma_i \in S$) have sizes (L_i^{σ}): TT streams (2) are $65Bytes$, and the BE stream is $35Bytes$.

Stream redundancy further enhances network reliability. The first and third streams are designated as non-redundant, indicated by a stream redundancy status of $\Pi_i^{\sigma} = 0$. This means they follow a single predetermined path through the network without any alternative routes. In contrast, the second stream is classified as redundant, with a stream redundancy status of $\Pi_i^{\sigma} = 1$. Consequently, it possesses an alternative stream

that traverses a different path, providing an additional layer of fault tolerance. This redundancy ensures data delivery even in the event of link failures or congestion on the primary path, contributing to a more robust and reliable network operation.

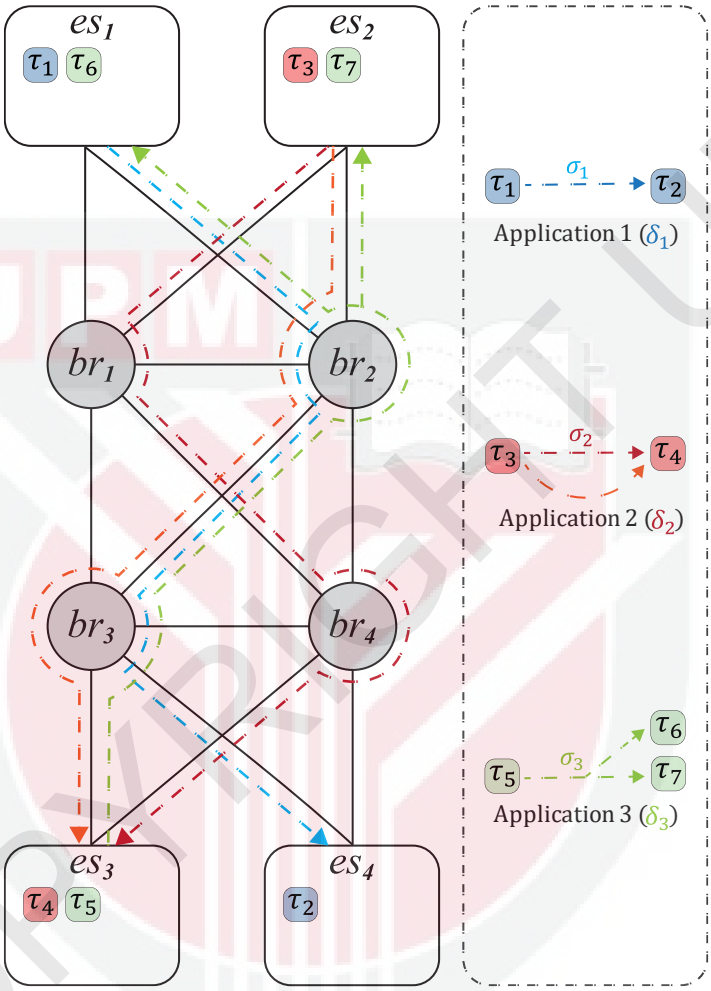

Figure 3.6 : The Network Topology of the Exemplary Problem Instance Illustrates the Route Taken by Each Stream

Applications δ_1 and δ_2 have a Worst-Case Execution Time (WCET) ω_j^τ of $35\mu s$, while application δ_3 has a WCET of $25\mu s$. All tasks and streams have a fixed period of $1000\mu s$, resulting in a hyperperiod (HP) of $1000\mu s$. Table 3.5 summarizes application, stream, and task parameters of the redundancy-considering exemplary problem instance.

Table 3.5 : The Applications, Streams, and Task Parameters of the Redundancy-Considering Exemplary Problem Instance

Δ	Γ_k^δ	T	ω_j^τ	Γ_j^τ	S	ρ_i^σ	L_i^σ	Γ_i^σ	Π_i^σ
δ_1	1000	τ_1, τ_2	35	1000	σ_1	TT	65	1000	0
δ_2	1000	τ_3, τ_4	35	1000	σ_2	TT	65	1000	1
δ_3	1000	τ_5, τ_6, τ_7	25	1000	σ_3	BE	35	1000	0

In alignment with the established period of 1000µs, four distinct gating times are defined, representing the moments when gate statuses undergo transitions. These transitions, as detailed in Figure 3.7 and the Gate Control List (GCL) entry, orchestrate the flow of data across the network by controlling access to bridge queues.

Four gating times, aligned with the period, control data flow by managing bridge queue access, as shown in the GCL entry in Figure 3.7. At $0\mu s$, TT gates (g_0 and g_1,) open (1), prioritizing TT traffic. At $450\mu s$, these gates close (0). BE gates (g_6 and g_7) open (1) at $650\mu s$ after the TT transmission window and close (0) at 1000µs, completing the cycle. The QGT is initially $0\mu s$ for all queues, indicating initial closure. Subsequent QGT values are $450\mu s$, $650\mu s$, and 1000µs for all queues, synchronizing their opening and closing times with the respective gating times.

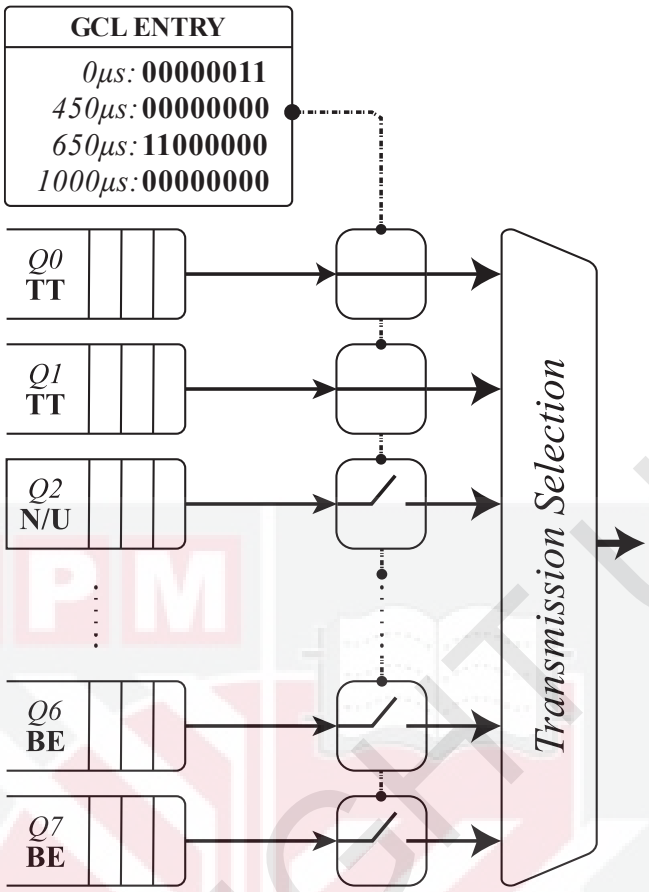

Figure 3.7 : A Simplified TSN Bridge in the Exemplary Problem Instance Features Queues and Gates, along with a Gate Control List (GCL) Entry

Figure 3.8 details the transmission schedule, illustrating traffic types and stream redundancy. Figure 3.8(a) shows Application 1 using a unicast TT stream, following a single path. Application 2 (Figure 3.8(b)) employs a redundant TT stream (S) with a redundancy status of $\Pi_i^\sigma = 1$. The original stream (σ_{2A}) and its redundant counterpart (σ_{2B}) follow distinct paths, with σ_{2B} transmitting after a temporal shift. This ensures delivery even if one path is disrupted; the listener task triggers upon arrival of either stream, discarding the other. Finally, Application 3 (Figure 3.8(c)) uses a multicast BE stream, transmitting from one talker to two listeners simultaneously.

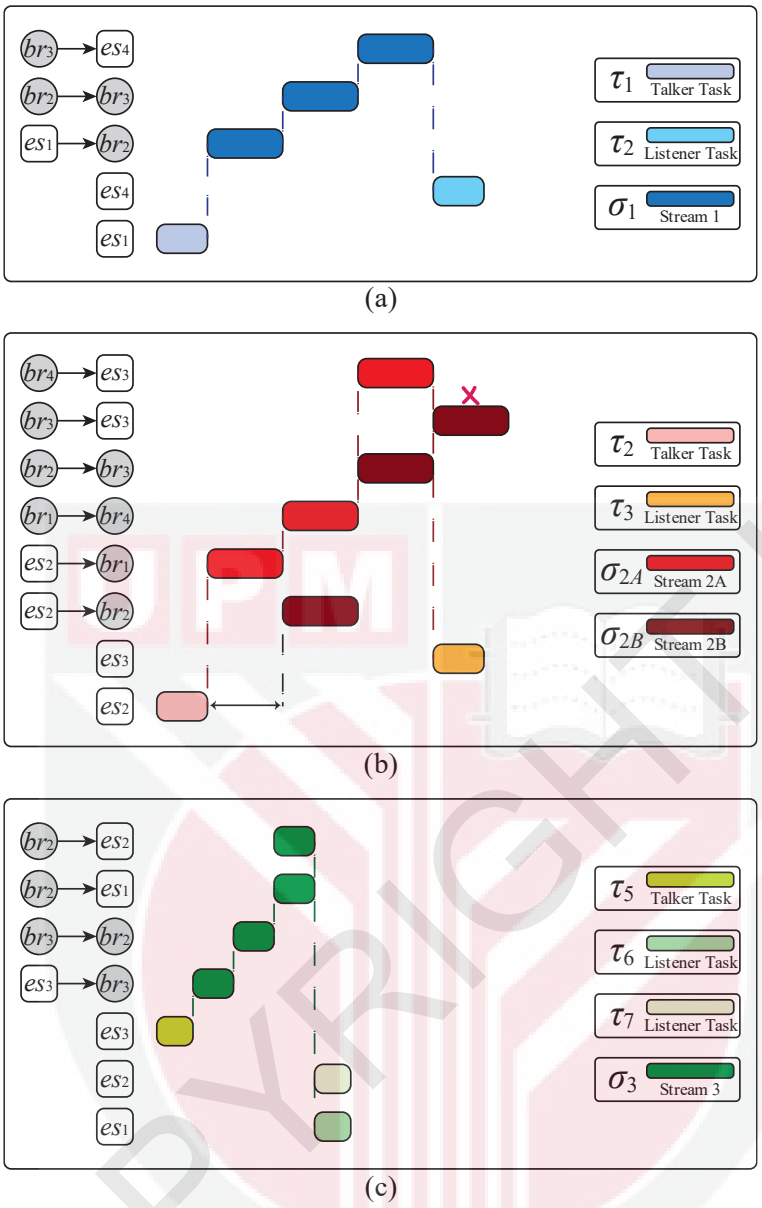

Figure 3.8 : The Network Schedule for the Exemplary Problem Instance Includes: (a) Transmission of Application 1 (δ_1) for a Unicast Time-Triggered (TT) Stream, (b) Transmission of Application 2 (δ_2) for a Unicast Redundant TT Stream, and (c) Transmission of Application 3 (δ_3) for a Multicast Best-Effort (BE) Stream

3.5 Optimization Framework

Finding the optimal schedule and route for data in time-sensitive networks is like solving a complex puzzle, especially as networks become larger and more intricate. These challenges are classified as "NP-hard" due to their similarity to the Bin-Packing

problem, as referenced in [46] and [57], meaning finding the perfect solution for large networks can be computationally impossible within a reasonable timeframe.

To address the complex problem of scheduling and routing tasks in networks, a new framework called the "Optimized Hybrid Deterministic Scheduling and Routing" (OHDSR) is introduced. This method uses Constraint Programming (CP) to find the best way to schedule and route tasks. CP is a tool that helps us find the best solution to complex problems by setting rules (constraints) and then finding the best way to solve them. OHDSR can handle two types of traffic: Time-Triggered (TT) and Best-Effort (BE). TT traffic needs to be delivered on time. This dual-traffic capability highlights the hybrid nature of the OHDSR framework. BE traffic is less strict about timing, so it can be more flexible. OHDSR balances these two types to use resources efficiently. OHDSR uses a fixed schedule for TT traffic to ensure it's delivered on time. For BE traffic, it uses CP's flexibility to find the best way to route it. This combination helps find good solutions for large and complex networks.

The OHDSR framework has been enhanced to develop the "Optimized Hybrid Deterministic Scheduling and Routing Plus" (OHDSR+) framework. OHDSR+ incorporates redundancy to improve the reliability of scheduling and routing. OHDSR+ differs from the original OHDSR by helping to avoid potential network failures, such as broken connections or failed devices. It includes these possibilities in its planning. By planning for potential problems, OHDSR+ can find solutions that are not only efficient but also reliable, even when things go wrong in the network. This makes it more useful for real-world networks that can be unpredictable.

3.6 Optimization Constraints

Optimization solvers identify optimal solutions by evaluating numerous possibilities against an objective function. This search is constrained by boundaries, ensuring feasibility. These constraints are categorized into six groups: routing, priority scheduling, stream scheduling, task scheduling, stream path reliability, and stream schedule reliability constraints. Each category influences the solution space and guides the solver.

3.6.1 Routing Constraints

Stream transmission requires routing constraints to ensure delivery from the talker to listeners. These constraints define permissible paths through intermediate nodes (bridges). The design and implementation of these routing constraints draw inspiration from established research in the field, notably the works presented in references [123] and [57]. A stream's network path is determined by identifying the sequence of traversed nodes, specifically by pinpointing the successor node for each node along the route. For a stream σ_i and a node v_a on its path, the successor node is denoted as $\mathrm{Z}_{v_a}^{\sigma_i}$.

A reverse route approach constructs successor nodes in a hierarchical, tree-like manner, starting from receiver nodes and progressing toward the source. The determination of $\mathrm{Z}_{v_a}^{\sigma_i}$ follows a set of rules based on the node's position relative to the stream's path:

- If v_a is the talker initiating the stream, then its successor is naturally itself, represented as $\mathrm{Z}_{v_a}^{\sigma_i} = v_a$.

- When v_a lies along the stream's path, its successor becomes the next node it encounters, denoted as $\mathrm{Z}_{v_a}^{\sigma_i} = v_b$.
- Conversely, if v_a is not part of the stream's journey, it has no successor, indicated by $\mathrm{Z}_{v_a}^{\sigma_i} = nil$. S

This systematic evaluation of successor nodes for every node and stream provides a clear and complete picture of the routes each stream traverses within the network.

To illustrate the concept of successor nodes and path lengths, let's revisit the example presented in Section 3.4.2, specifically focusing on stream σ_1, which follows the path: $[es_1 \longrightarrow br_2 \longrightarrow br_3 \longrightarrow es_4]$. In this scenario, the successor node $\mathrm{Z}_{v_a}^{\sigma_1}$ takes on different values depending on the position of v_a:

- If v_a is the talker node (es_1) or the bridge node br_2, then $\mathrm{Z}_{v_a}^{\sigma_1}$ is br_2.
- For v_a positioned at br_3, the successor becomes es_4.
- And for v_a at the destination es_4, the successor is 'nil' as there are no further nodes in the path.
- For any node not part of the path of σ_1, such as es_2, es_3, br_1, and br_4, the successor node is designated as 'nil', signifying the absence of a subsequent node on the stream's route.

Figure 3.9 provides a visual representation of this concept for clarity. In addition to successor nodes, the length of the path from any node v_a to the talker node of stream σ_1, denoted as $\Upsilon_{v_a}^{\sigma_i}$, is also considered. The permissible values for this path length range from 0 to the maximum number of bridges plus 1 ($|BR| + 1$). This accounts for

scenarios where the node is the talker itself (path length 0) or located at the farthest possible point in the network relative to the talker.

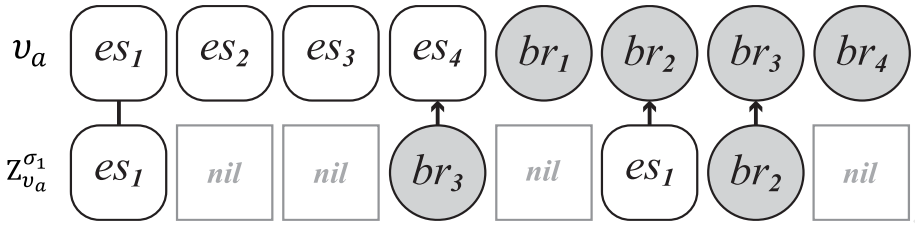

Figure 3.9 : The Value of $Z_{v_a}^{\sigma_i}$ for Stream σ_1 from the Exemplary Problem Instance

The specific constraints governing routing are mathematically formulated in equations 3.1 through 3.5, providing a rigorous framework for ensuring efficient and valid stream paths within the network.

$$\forall \sigma_i \in S,\ \forall v_a \in \mathcal{V}\ except\ \{es\langle \tau_{talker}^{\sigma_i} \rangle\}:$$

$$\left(Z_{v_a}^{\sigma_i} \neq nil\right) \Longrightarrow \left(\Upsilon_{v_a}^{\sigma_i} = \Upsilon_{Z_{v_a}^{\sigma_i}} + 1\right) \tag{3.1}$$

Equation (3.1) plays a critical role in maintaining the integrity of stream paths by preventing the formation of cycles. It essentially states that for any node v_a with a successor along the path of a stream (meaning $Z_{v_a}^{\sigma_i}$ is not ' nil '), the path length to v_a ($\Upsilon_{v_a}^{\sigma_i}$) must be exactly one unit greater than the path length to its successor node ($\Upsilon_{Z_{v_a}^{\sigma_i}}$). This constraint ensures a logical progression along the path, where each subsequent node is one step further away from the talker node compared to its predecessor. By enforcing this rule, the equation effectively eliminates the possibility of loops or cycles within the stream's route, guaranteeing a direct and efficient path from the talker to the listener.

$$\forall \sigma_i \in S,\ \ \forall v_a, v_b \in \mathcal{V}:$$

$$Z_{v_b}^{\sigma_i} = nil \Leftrightarrow Z_{v_a}^{\sigma_i} \neq v_b \tag{3.2}$$

Equation (3.2) reinforces the interconnectedness of nodes within a stream's path by establishing a reciprocal relationship between successors and predecessors. It dictates that if a node has a successor along the path of a stream ($Z_{v_b}^{\sigma_i} = nil$) (implying it's not the final destination), then it must also have a predecessor ($Z_{v_a}^{\sigma_i} \neq v_b$) (a node it arrived from). This principle stems from the reverse route approach employed to determine stream paths, ensuring that each node along the route is linked to both a preceding and succeeding node. By enforcing this bidirectional connection, the equation eliminates the possibility of "loose ends" or isolated nodes within the path, guaranteeing a continuous and well-defined route for the stream from its source to its destination.

$$\forall \sigma_i \in S,\ \ \forall v_a \in es\langle \tau_{listener}^{\sigma_i} \rangle,\ \ where\ \tau_{listener}^{\sigma_i} \in T_{listener}^{\sigma_i}:$$

$$Z_{v_a}^{\sigma_i} \neq nil \tag{3.3}$$

Equation (3.3) focuses specifically on the listener end-systems, which are the destinations of a stream. It mandates that every listener node ($\forall v_a \in es\langle \tau_{listener}^{\sigma_i} \rangle$) associated with a stream must have a defined successor ($Z_{v_a}^{\sigma_i} \neq nil$). This might seem counterintuitive at first, as listeners are typically considered the endpoints of a stream's journey. However, this constraint acknowledges the possibility of multiple listeners for a single stream and ensures that each listener is integrated into the stream's path with a clear predecessor node. This guarantees a structured and well-defined route for the stream, even when it needs to reach multiple destinations.

$$\forall \sigma_i \in S:$$

$$Z^{\sigma_i}_{es\langle \tau^{\sigma_i}_{talker} \rangle} = es\langle \tau^{\sigma_i}_{talker} \rangle,$$

$$Y^{\sigma_i}_{es\langle \tau^{\sigma_i}_{talker} \rangle} = 0 \tag{3.4}$$

Equation (3.4) establishes the starting point for any stream's journey within the network. It explicitly states that for each stream σ_i, the talker node, denoted as $es\langle \tau^{\sigma_i}_{talker} \rangle$, must identify itself as its own successor. This signifies the initiation of the stream's path from the talker node. Furthermore, the equation sets the path length at the talker's position to zero ($Y^{\sigma_i}_{es\langle \tau^{\sigma_i}_{talker} \rangle} = 0$), signifying the beginning of the path with no preceding nodes. This constraint ensures a clear and unambiguous starting point for each stream, providing a foundation for the subsequent routing decisions and path calculations.

$$\forall \sigma_i \in S, \ \forall v_a \in ES \ except \ \{es\langle \tau^{\sigma_i}_{listener} \rangle \ and \ es\langle \tau^{\sigma_i}_{talker} \rangle\},$$

$$where \ \tau^{\sigma_i}_{listener} \in T^{\sigma_i}_{listener}:$$

$$Z^{\sigma_i}_{v_a} = nil \tag{3.5}$$

Equation (3.5) addresses the role of end-system nodes that are not involved in a particular stream's transmission. It states that for any end-system node that does not function as a listener ($\tau^{\sigma_i}_{listener}$) or a talker ($\tau^{\sigma_i}_{talker}$) for a given stream σ_i, its successor node will be designated as ' nil '. This essentially excludes these uninvolved end-systems from the stream's path, as they neither initiate the stream nor serve as its destination. It's important to note that this constraint applies specifically to end-system nodes and does not affect bridge nodes, which can still serve as intermediate points

along the stream's route even if they are not directly associated with the stream's origin or endpoint.

3.6.2 Priority Scheduling Constraints

The priority scheduling constraint comes into play at each bridge within the network, denoted as $br_n \in BR$. As streams of diverse types (ρ_i^{σ}) traverse these bridges, they are categorized and queued based on their respective types. As elaborated in section 3.3, each bridge is equipped with a set of eight queues, represented as $q_{\phi}^{br_n} \in Q^{br_n}$, each associated with a distinct type ρ_{ϕ}^{q}. This arrangement facilitates organized stream management, as streams sharing the same type as a particular queue are directed to that specific queue. This type-based queuing system lays the groundwork for prioritizing stream transmission based on their assigned types and the corresponding queue priorities.

Each queue within a bridge operates using two Queue Gating Time (QGTs), denoted as $\kappa_{\varphi}^{q}\boxed{\psi}$, which dictate the open and closed states of the queue. These QGTs, representing moments in time, control the flow of streams by determining when the queue is available for transmission (open, represented by 1) and when it is not (closed, represented by 0). The values of these QGTs are derived from the gating time associated with the bridge, denoted as $t_{\varphi}^{br_n} \in \Phi^{br_n}$. This gating time serves as a reference point from which the specific open and closed times for each queue are calculated. This mechanism ensures a structured and synchronized approach to stream transmission, preventing congestion and optimizing the utilization of network resources.

$$\forall \sigma_i \in S,\ \forall br_n \in BR,\ \forall q_\phi^{br_n} \in Q^{br_n},$$

$$t_\varphi^{br_n} \in \Phi^{br_n},\ \rho_i^\sigma == \rho_\phi^q:$$

$$\eta_{br_n}^{\sigma_i} \geq \kappa_\varphi^{q^{br_n}}[1] \tag{3.6}$$

$$\zeta_{br_n}^{\sigma_i} \leq \kappa_\varphi^{q^{br_n}}[0] \tag{3.7}$$

Equations (3.6) and (3.7) plays a crucial role in coordinating stream transmission with the queue gating mechanism at each bridge. It defines the permissible time window for both the offset (start time) ($\eta_{br_n}^{\sigma_i}$) and end time of each stream passing through a bridge ($\zeta_{br_n}^{\sigma_i}$). The constraint dictates that the offset of a stream σ_i must be greater than or equal to the Queue Gating Time (QGT) when the queue is in an "open" state (status = 1) ($\kappa_\varphi^{q^{br_n}}[1]$). This ensures that the stream begins transmission only when the queue is ready to forward it. Conversely, the end time of the same stream must be less than or equal to the QGT when the queue is "closed" (status = 0) ($\kappa_\varphi^{q^{br_n}}[0]$), guaranteeing that the stream's transmission concludes before the queue enters a non-forwarding state. It's important to note that this constraint is specific to bridges and does not apply to streams at end-systems ($es_m \in ES$) or links ($\upsilon_a, \upsilon_b) \in \mathcal{E}$, as these components do not incorporate queues and gating mechanisms in their design.

3.6.3 Stream Scheduling Constraints

In this section, the critical aspect of stream scheduling constraints is explored. These constraints play a pivotal role in optimizing the transmission of streams across the network by establishing rules and guidelines for scheduling their delivery. By adhering to these constraints, our optimization framework gains the ability to identify the most

efficient and effective schedule for stream transmission, ensuring timely delivery while maximizing network utilization. The following constraints will provide a detailed framework for achieving optimal stream scheduling.

$$\forall \sigma_i \in S,\ \ \forall (\upsilon_a, \upsilon_b) \in \mathcal{E},\ \ \forall (\upsilon_a, \upsilon_b) \in \mathbb{R}_i^{\sigma}:$$

$$\varepsilon_{(\upsilon_a,\upsilon_b)}^{\sigma_i} = \left\lceil \frac{L_i^{\sigma}}{\Omega_{(\upsilon_a,\upsilon_b)}} \right\rceil \tag{3.8}$$

The initial step in stream scheduling involves determining the precise duration each stream will occupy a given link. Equation (3.8) provides the formula for this calculation, expressing the transmission time as a function of the stream's size and the link's speed. Specifically, the stream size (L_i^{σ}), measured in bytes, is divided by the link speed ($\Omega_{(\upsilon_a,\upsilon_b)}$), expressed in megabits per second (Mbps). This division yields the transmission duration ($\varepsilon_{(\upsilon_a,\upsilon_b)}^{\sigma_i}$), measured in microseconds (μs), representing the time required for the entire stream to traverse the link. This calculated duration serves as a fundamental building block for constructing an efficient stream schedule, ensuring that link capacities are respected and transmission times are accurately accounted for.

$$\forall \sigma_i \in S,\ \ \forall (\upsilon_a, \upsilon_b) \in \mathcal{E},\ \ \forall (\upsilon_a, \upsilon_b) \in \mathbb{R}_i^{\sigma}:$$

$$\zeta_{(\upsilon_a,\upsilon_b)}^{\sigma} = \eta_{(\upsilon_a,\upsilon_b)}^{\sigma} + \varepsilon_{(\upsilon_a,\upsilon_b)}^{\sigma} \tag{3.9}$$

Following the calculation of stream durations, Equation (3.9) enforces a fundamental relationship between a stream's start time, duration, and end time. It dictates that the end time of each stream ($\zeta_{(\upsilon_a,\upsilon_b)}^{\sigma}$) must be precisely equal to the sum of its offset (start time) ($\eta_{(\upsilon_a,\upsilon_b)}^{\sigma}$) and its duration ($\varepsilon_{(\upsilon_a,\upsilon_b)}^{\sigma}$). This constraint applies universally to all streams ($\sigma_i \in S$) across every link they traverse along their designated paths ($\forall (\upsilon_a, \upsilon_b) \in \mathbb{R}_i^{\sigma}$). By enforcing this consistency, stream schedules are ensured to be

logically sound and temporally coherent, preventing overlaps and conflicts while maintaining a clear timeline for each stream's transmission.

$$\forall \sigma_i \in S,\ \ \forall (\upsilon_b, \upsilon_c) \in \mathcal{E} \cap \mathbb{R}_i^\sigma,\ \ \upsilon_a = \mathrm{Z}_{\upsilon_b}^\sigma:$$

$$\zeta_{(\upsilon_a,\upsilon_b)}^\sigma \leq \eta_{(\upsilon_b,\upsilon_c)}^\sigma \tag{3.10}$$

Equation (3.10) plays a crucial role in maintaining the order and flow of streams as they traverse multiple links within their designated paths. It focuses on any two consecutive links, denoted as υ_a, υ_b and υ_b, υ_c, that are part of a stream's route ($\mathbb{R}_i^\sigma$). The constraint dictates that the stream's offset (start time) ($\eta_{(\upsilon_b,\upsilon_c)}^\sigma$) at the second link (υ_b, υ_c) must be greater than or equal to the end time of the stream ($\zeta_{(\upsilon_a,\upsilon_b)}^\sigma$) at the first link (υ_a, υ_b). This ensures a logical sequence of transmission, where the stream completes its journey on one link before initiating transmission on the next. By enforcing this order, the equation prevents temporal overlaps and ensures a smooth and efficient flow of the stream along its designated path.

$$\forall \sigma_1, \sigma_2 \in S, \sigma_1 \neq \sigma_2, \forall (\upsilon_a, \upsilon_b) \in \mathbb{R}_1^\sigma \cap \mathbb{R}_2^\sigma,$$

$$\forall \alpha \in \left\{0, \cdots, \frac{lcm(\Gamma_1^\sigma, \Gamma_2^\sigma)}{\Gamma_1^\sigma}\right\},$$

$$\forall \beta \in \left\{0, \cdots, \frac{lcm(\Gamma_1^\sigma, \Gamma_2^\sigma)}{\Gamma_2^\sigma}\right\}:$$

$$\left(\left(\alpha \times \Gamma_1^\sigma + \zeta_{(\upsilon_a,\upsilon_b)}^{\sigma_1}\right) \leq \left(\beta \times \Gamma_2^\sigma + \eta_{(\upsilon_a,\upsilon_b)}^{\sigma_2}\right)\right) \vee$$

$$\left(\left(\beta \times \Gamma_2^\sigma + \zeta_{(\upsilon_a,\upsilon_b)}^{\sigma_2}\right) \leq \left(\alpha \times \Gamma_1^\sigma + \eta_{(\upsilon_a,\upsilon_b)}^{\sigma_1}\right)\right) \tag{3.11}$$

Equation (3.11) ensures non-overlapping transmissions for two distinct streams, σ_1 and σ_2, on a shared link (υ_a, υ_b) that lies within their respective routes ($\mathbb{R}_1^\sigma$ and $\mathbb{R}_2^\sigma$).

The condition leverages the periods Γ_1^σ and Γ_2^σ of the streams, along with their least common multiple (lcm), to define the potential repetition intervals. For each repetition, represented by α for σ_1 and β for σ_2, the equation ensures that the start and end times of the transmissions on the shared link do not overlap. This is achieved by enforcing that either σ_1's transmission completes before σ_2 starts or vice versa, considering their respective offsets $\eta^{\sigma_1}_{(\upsilon_a,\upsilon_b)}$ and $\eta^{\sigma_2}_{(\upsilon_a,\upsilon_b)}$. By preventing simultaneous use of the link, the equation avoids conflicts, congestion, and contention, ensuring fair and efficient allocation of resources. This deterministic scheduling improves the predictability, reliability, and overall performance of the network, making it particularly suited for time-sensitive applications.

$$\forall \sigma_1, \sigma_2 \in S, \sigma_1 \neq \sigma_2, \forall (\upsilon_b, \upsilon_c) \in \mathbb{R}_1^\sigma \cap \mathbb{R}_2^\sigma,$$

$$\forall (\upsilon_{a1}, \upsilon_b) \in \mathbb{R}_1^\sigma, \qquad \forall (\upsilon_{a2}, \upsilon_b) \in \mathbb{R}_2^\sigma,$$

$$\upsilon_{a1} = \mathrm{Z}_{\upsilon_b}^{\sigma_1}, \ \upsilon_{a2} = \mathrm{Z}_{\upsilon_b}^{\sigma_2},$$

$$\forall \alpha \in \left\{0, \cdots, \frac{lcm(\Gamma_1^\sigma, \Gamma_2^\sigma)}{\Gamma_1^\sigma}\right\}, \ \forall \beta \in \left\{0, \cdots, \frac{lcm(\Gamma_1^\sigma, \Gamma_2^\sigma)}{\Gamma_2^\sigma}\right\}:$$

$$\forall \left(\left(\alpha \times \Gamma_2^\sigma + \eta^{\sigma_2}_{(\upsilon_b,\upsilon_c)}\right) \leq \left(\beta \times \Gamma_1^\sigma + \eta^{\sigma_1}_{(\upsilon_{a1},\upsilon_b)}\right)\right) \vee$$

$$\left(\left(\beta \times \Gamma_1^\sigma + \eta^{\sigma_1}_{(\upsilon_b,\upsilon_c)}\right) \leq \left(\alpha \times \Gamma_2^\sigma + \eta^{\sigma_2}_{(\upsilon_{a2},\upsilon_b)}\right)\right) \tag{3.12}$$

Equation (3.12) tackles a critical challenge in Time-Sensitive Networking (TSN). Ensuring the smooth and reliable transmission of frames, particularly when multiple streams converge at bridge nodes. Imagine a bridge node (υ_b or br_n) where two streams, σ_1 and σ_2, meet. Each stream originates from a different source node (υ_{a1} and υ_{a2}) and travels through separate links ($(\upsilon_{a1}, \upsilon_b)$and $(\upsilon_{a2}, \upsilon_b)$) before arriving at the bridge. From this bridge, both streams will share a common path onward, utilizing

the same link (υ_b, υ_c) as part of their respective routes ($\mathbb{R}_1^\sigma$ and $\mathbb{R}_2^\sigma$). To prevent collisions and potential frame loss at the bridge, Equation (3.12) enforces a "frame isolation" principle. This principle, originally proposed by Craciunas et al. [124] and now widely adopted in TSN, dictates that the frames arriving from the two different streams must not overlap in time. In simpler terms, the frame from stream σ_1 must completely arrive at the bridge node before the frame from stream σ_2 starts to arrive, and vice versa. This constraint ensures sufficient processing time for each frame at the bridge, preventing delays and maintaining data integrity, thus enhancing transmission resilience and overall TSN network robustness.

3.6.4 Task Scheduling Constraints

Tasks, the core units of work within an application, are linked to data streams. Efficient task scheduling is crucial for application performance, necessitating task scheduling constraints. These constraints dictate task timing and order to prevent conflicts, ensure data availability, and maximize efficiency, contributing to smooth application operation.

$$\forall \tau_j \in T:$$

$$\zeta_{\upsilon_a}^{\tau} = \eta_{\upsilon_a}^{\tau} + \omega_j^{\tau} \tag{3.13}$$

Equation (3.13) establishes a clear relationship between a task's start time, execution time, and end time. It focuses on calculating the end time for each task τ_j executed at a specific node υ_a. Each task is characterized by its worst-case execution time (ω_j^{τ}), which represents the maximum duration required for the task to complete its operation under any circumstance. The equation then dictates that the end time of the task ($\zeta_{\upsilon_a}^{\tau}$)

is determined by adding this worst-case execution time (ω_j^τ) to the task's offset (start time) ($\eta_{\upsilon_a}^\tau$). This calculation provides a reliable estimate of when the task will finish its execution, aiding in planning and coordinating subsequent tasks and ensuring efficient utilization of processing resources.

$$\forall \tau_j \in T,\ \forall \sigma_i \in S,\ \tau_j = \tau_{talker}^{\sigma_i},$$
$$\forall (\upsilon_a, \upsilon_b) \in \mathcal{E} \cap \mathbb{R}_i^\sigma,\ \upsilon_a == es_j^\tau:$$
$$\zeta_{\upsilon_a}^\tau \leq \eta_{(\upsilon_a,\upsilon_b)}^\sigma \tag{3.14}$$

Tasks and streams exhibit interdependencies, requiring careful temporal coordination for smooth operation. Tasks may depend on stream data, and streams may depend on task completion. Equation (3.14) addresses this coordination specifically for tasks at talker nodes and their corresponding outgoing streams. The constraint dictates that any task τ_j executed at a talker node υ_a must be fully completed before the associated outgoing stream σ_i can begin its transmission through the outgoing link (υ_a, υ_b),. This implies that the offset (start time) of the stream ($\eta_{(\upsilon_a,\upsilon_b)}^\sigma$) must be greater than or equal to the end time of the task ($\zeta_{\upsilon_a}^\tau$). This sequencing ensures that the stream carries the most up-to-date data generated by the task, preventing inconsistencies and ensuring the integrity of information flowing through the network.

$$\forall \tau_j \in T,\ \forall \sigma_i \in S,\ \tau \in T_{listner}^{\sigma_i},$$
$$\forall (\upsilon_a, \upsilon_b) \in \mathcal{E} \cap \mathbb{R}_i^\sigma,\ \upsilon_b == es_j^\tau:$$
$$\zeta_{(\upsilon_a,\upsilon_b)}^\sigma \leq \eta^\tau \tag{3.15}$$

Expanding on the concept of task-stream coordination, Equation (3.15) focuses on the receiving end of stream transmission, specifically addressing the relationship between

incoming streams and tasks at listener nodes. The constraint emphasizes that tasks at a listener node must not commence execution before the arrival of the data they depend on, which is carried by the incoming stream. More precisely, for a stream σ_i arriving at a listener node v_b via link (v_a, v_b), the offset (start time) of any task (η^τ) at node v_b that relies on this stream's data must be greater than or equal to the end time of the stream's transmission on the link ($\zeta^\sigma_{(v_a,v_b)}$). This sequencing ensures that the task has access to the complete and up-to-date data it requires before initiating its execution, preventing errors and ensuring the correct functioning of the application.

$$\forall \tau_1, \tau_2 \in T, \quad \tau_1 \neq \tau_2,$$
$$\forall \alpha \in \left\{0, \cdots, \frac{lcm(\Gamma_1^\tau, \Gamma_2^\tau)}{\Gamma_1^\tau}\right\}, \quad \forall \beta \in \left\{0, \cdots, \frac{lcm(\Gamma_1^\tau, \Gamma_2^\tau)}{\Gamma_2^\tau}\right\},$$
$$\forall \left(\left(\alpha \times \Gamma_1^\tau + \zeta_{v_a}^{\tau_1}\right) \leq \left(\beta \times \Gamma_2^\tau + \eta_{v_a}^{\tau_2}\right)\right) \vee$$
$$\left(\left(\beta \times \Gamma_2^\tau + \zeta_{v_a}^{\tau_2}\right) \leq \left(\alpha \times \Gamma_1^\tau + \eta_{v_a}^{\tau_1}\right)\right) \tag{3.16}$$

Equation (3.16) ensures conflict-free scheduling of multiple tasks, τ_1 and τ_2, on the same end-system node v_a. It addresses the potential overlap in execution times of distinct tasks by enforcing that one task must fully complete its execution before the other begins. Using the periods Γ_1^τ and Γ_2^τ of the tasks, along with their least common multiple (lcm), the equation examines all possible repetition intervals. For each repetition, represented by α for τ_1 and β for τ_2, the constraint ensures that either τ_1's execution window, starting at $(\alpha \times \Gamma_1^\tau + \zeta_{v_a}^{\tau_1})$, finishes before τ_2 starts at $(\beta \times \Gamma_2^\tau + \eta_{v_a}^{\tau_2})$, or vice versa. This non-overlapping condition eliminates competition for processing resources on v_a, ensuring that each task receives dedicated and uninterrupted execution time. By upholding this strict temporal isolation, the equation

enhances the predictability, stability, and efficiency of the end-system, which is crucial for maintaining the performance of time-sensitive applications.

3.6.5 Stream Path Reliability Constraints

In the realm of TT stream transmission, reliability is of utmost importance. To address this, a redundancy mechanism is specifically implemented for OHDSR+ to bolster the resilience of critical streams. Each stream is assigned a redundancy status, denoted as Π_i^σ. When $\Pi_i^\sigma = 1$, it signifies that the stream requires redundancy, triggering the application of the constraint defined in (3.17). This constraint guarantees path diversity between the original stream, σ_{iA}, and its redundant counterpart, σ_{iB}.

$$\forall \sigma_i \in S,\ \ \Pi_i^\sigma = 1,\ \ \forall v_a \in \mathcal{V}\ except\ \{es\langle \tau_{talker}^{\sigma_i} \rangle\}:$$

$$Z_{v_a}^{\sigma_{iA}} \neq Z_{v_a}^{\sigma_{iB}} \tag{3.17}$$

The core principle behind this constraint is to ensure that at every single node within the network, the successor ($Z_{v_a}^{\sigma_{iB}}$) chosen by the redundant stream, σ_{iB}, is different from the one ($Z_{v_a}^{\sigma_{iA}}$) chosen by the original stream, σ_{iA}. Enforcing distinct paths for streams mitigates the risk of shared failures, such as node or link outages. Redundant streams, traversing alternative routes, provide resilience to disruptions in the primary path, ensuring reliable transmission of critical data.

3.6.6 Stream Schedule Reliability Constraints

To enhance the resilience of stream transmission against potential disruptions, the stream path reliability constraint in Equation (3.17) was previously introduced specifically for OHDSR+, ensuring that original and redundant streams traverse

different paths through the network. Building upon this foundation, Equation (3.18) introduces an additional layer of protection by enforcing a temporal separation between the original and redundant streams. This temporal shift acts as a buffer, mitigating the impact of transient errors or network congestion.

$$\forall \sigma_i \in S,\ \ \Pi_i^{\sigma} = 1,$$

$$\forall (v_a, v_b), (v_a, v_c) \in \mathcal{E} \cap \mathbb{R}_i^{\sigma},\ \ v_a == es\langle \tau_{talker}^{\sigma_i} \rangle:$$

$$\eta_{(v_a, v_c)}^{\sigma_{iB}} \geq \zeta_{(v_a, v_b)}^{\sigma_{iA}} \tag{3.18}$$

The constraint dictates that the offset (start time) ($\eta_{(v_a, v_c)}^{\sigma_{iB}}$) of the redundant stream (σ_{iB}) on a particular link (v_a, v_b) must be greater than or equal to the end time ($\zeta_{(v_a, v_b)}^{\sigma_{iA}}$) of the original stream (σ_{iA}) on a different link (v_a, v_c). Importantly, both of these links connect the same talker end-system (v_a) to two distinct bridge nodes (v_b) and (v_c). By enforcing this temporal separation, a safety net is created where the redundant stream begins its journey only after the original stream has progressed through a portion of the network. This minimizes the risk of simultaneous disruptions affecting both streams and ensures a higher likelihood of successful data delivery.

$$\forall \sigma_i \in S,\ \ \Pi_i^{\sigma} = 1,\ \ \forall \tau_j \in T,\ \ \tau_j \in T_{listner}^{\sigma_i},$$

$$\forall (v_a, v_c), (v_b, v_c) \in \mathcal{E} \cap \mathbb{R}_i^{\sigma},\ \ v_c == es_j^{\tau}:$$

$$\begin{cases} \zeta_{(v_a, v_c)}^{\sigma_{iA}} \leq \eta^{\tau}, & if\ \zeta_{(v_a, v_c)}^{\sigma_{iA}} \leq \zeta_{(v_b, v_c)}^{\sigma_{iB}} \\ \zeta_{(v_b, v_c)}^{\sigma_{iB}} \leq \eta^{\tau}, & if\ \zeta_{(v_b, v_c)}^{\sigma_{iB}} < \zeta_{(v_a, v_c)}^{\sigma_{iA}} \end{cases} \tag{3.19}$$

In scenarios where redundant streams are employed to enhance reliability ($\Pi_i^{\sigma} = 1$) and a temporal shift is introduced between the original and redundant streams, it's essential to manage the initiation of listener tasks effectively. Equation (3.19)

addresses this by ensuring that the listener task begins execution upon receiving either the original or redundant stream, whichever arrives first, while disregarding the other. The constraint dictates that the listener task's offset (start time) (η^{τ}) is set to occur after the end time of the first arriving stream at the listener node. For instance, if the original stream (σ_{iA}) reaches the listener node (v_c) earlier than the redundant stream (σ_{iB}), the listener task will commence upon the completion ($\zeta^{\sigma_{iA}}_{(v_a,v_c)}$) of the transmission of σ_{iA} on the link connecting the listener to the last bridge (v_a) it traversed. Conversely, if the redundant stream (σ_{iB}) arrives first, the listener task will start upon its completion ($\zeta^{\sigma_{iB}}_{(v_b,v_c)}$), effectively ignoring the later arrival of the original stream (σ_{iA}). This approach ensures that the listener task is not unnecessarily delayed, utilizing the first available stream's data to commence its execution while maintaining data integrity and consistency.

3.7 Objective Functions

Our optimization framework is designed to tackle the intricate task of managing stream routing and scheduling while incorporating the priorities associated with different stream types. To achieve this, two distinct objective functions are employed, each targeting a specific aspect of the optimization process:

- Routing Optimization: The first objective function focuses on optimizing routing decisions, aiming to identify the most efficient and reliable paths for streams based on network conditions, link capacities, and stream priorities.
- Scheduling Optimization: The second objective function concentrates on scheduling, aiming to determine the optimal timing and sequencing of stream transmissions and task executions, taking into account factors such as task dependencies, execution times, and stream deadlines.

By incorporating these two objective functions, our framework comprehensively addresses both the spatial (routing) and temporal (scheduling) aspects of stream and task management, leading to an optimized and well-coordinated system.

The primary objective of routing optimization within our framework, as expressed in Equation (3.20), is to minimize the overall distance traversed by all streams within the network. This is achieved by calculating the sum of the lengths of all stream paths and seeking to minimize this cumulative value. For each stream (σ_i), the calculation involves considering every node along its path and checking if a successor node exists (i.e., the node is not a terminal point). If a successor is present ($Z_{v_a}^{\sigma_i} != nil$), it contributes to the total count of nodes traversed by the stream, excluding the initial talker node. Minimizing the sum of individual stream path lengths across all streams promotes efficient network resource utilization, reduces delays, and enhances system performance.

$$min \sum_{\sigma_i \in S} \sum_{\forall v_a \in \mathcal{V} \; except \; \{es\langle \tau_{talker}^{\sigma_i} \rangle\}} (Z_{v_a}^{\sigma_i} != nil) \tag{3.20}$$

The core objective of scheduling optimization in our study, as formulated in Equation (21), is to minimize the total latency experienced by all transmissions within the network. Latency, in this context, refers to the end-to-end duration of a transmission, encompassing the time it takes for data to travel from its origin to its destination and be processed by the receiving task. As illustrated in Figure 3.10, the latency measurement begins at the initiation ($\eta^{\tau_{talker}}$) of the talker task (τ_{talker}) at the talker node ($es^{\tau_{talker}}$) and concludes with the completion ($\zeta^{\tau_{listner}}$) of the listener task ($\tau_{listner}$) at the listener node ($es^{\tau_{listener}}$). This includes stream transmission time across the network, along with queuing and processing delays at intermediate nodes.

Minimizing the sum of these latencies ensures timely data delivery, improving application responsiveness and overall performance.

$$min \sum_{\delta_k \in \Delta} (\zeta^{\tau_{listner}} - \eta^{\tau_{talker}}),$$

$$where\ \tau_{listner},\ \tau_{talker} \in T_k \tag{3.21}$$

Our application framework employs tasks and streams to exchange data between end-systems via intermediary bridges. Minimizing total latency (Equation 3.21) improves data transmission and processing time, directly supporting time-sensitive networking's goal of predictable delivery. Reduced latency enhances application responsiveness and efficiency, enabling real-time communication in critical systems.

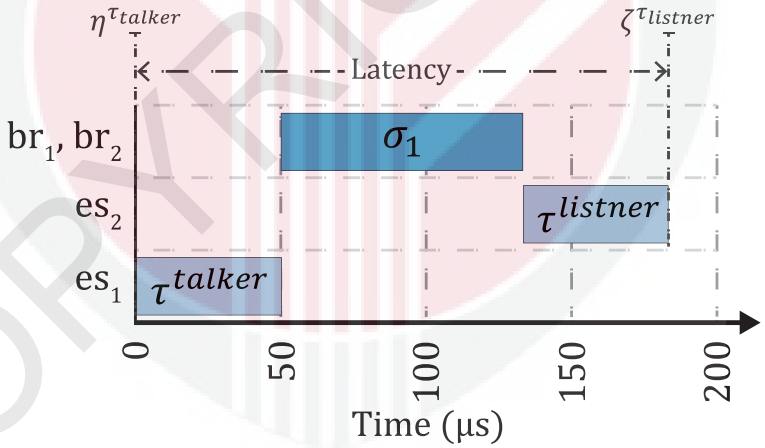

Figure 3.10 : The Value of $Z_{v_a}^{\sigma_i}$ for Stream σ_1 from the Exemplary Problem Instance

3.8 Experimental Setup

Constraint programming (CP) presents a robust and adaptable methodology for addressing optimization problems, leveraging a diverse array of tools accessible across

a multitude of programming languages. In the context of this study, the CP-Sat Solver [63], a pivotal component of the OR-Tools suite engineered by Google, was utilized. OR-Tools, employed here in version 9.6.2534, comprises an open-source compendium of optimization tools, furnishing a dependable and high-performance foundation for grappling with intricate problems characterized by complex constraints. The development and implementation of our models were facilitated by Visual Studio Code (VS Code) [125], specifically version 1.89.1, a ubiquitous, freely available, and open-source code editor developed by Microsoft, using the Python 3.11 [126] programming language.

To evaluate the performance and scalability of the proposed model, a diverse dataset encompassing a wide spectrum of problem instances was constructed. These instances were systematically varied in size and complexity, characterized by the number of bridges, end-systems, applications, streams, and tasks. The generation of these problem instances was achieved through a custom algorithm specifically developed for this purpose, leveraging the expressive capabilities of the NetworkX Python library [127]. This library, renowned for its graph manipulation and analysis functionalities, enabled the creation of intricate network topologies representative of real-world scenarios.

Our custom algorithm, implemented in Python, produces problem instances encoded in the Extensible Markup Language (XML) format [128]. This choice of XML facilitates interoperability, enabling seamless interaction and data exchange with our primary modeling language, Python. While the number of streams within each instance is treated as a deterministic input parameter, allowing for controlled

assessment of its impact, the remaining parameters, such as the number of bridges, end-systems, applications, and tasks, are assigned dynamically. This dynamic assignment, governed by carefully designed probability distributions, ensures diversity in instance characteristics, promoting a comprehensive evaluation of our model's performance across a broad range of operational conditions.

To guarantee the consistency and trustworthiness of the experimental outcomes, all tests were executed on a standardized hardware platform. This platform consisted of a system equipped with an Intel Core i7-11370H CPU, clocked at 3.30 GHz, and featuring an 8-core architecture with 8GB of RAM. This hardware configuration was deemed to provide a suitable and controlled environment for rigorously evaluating the performance and efficacy of our proposed optimization framework in conjunction with the chosen CP-Sat Solver. By standardizing the hardware environment, the potential influence of hardware variability on the experimental results is minimized, thus enhancing the reliability and generalizability of the findings.

3.9 Problem Instances

To ensure a comprehensive evaluation of the proposed framework, a diverse set of problem instances reflecting the characteristics commonly encountered in real-world automation and autonomous vehicle networks was created. The creation of problem instances involves generating a network topology, defining devices such as bridges and end systems, and configuring applications with their associated tasks and streams.

3.9.1 Procedure for Instances Creation

This process is automated using a Python [126] script that leverages several libraries to handle graph creation, visualization, and XML [128] file generation. The script begins by importing the necessary libraries: NetworkX (networkx) [127] for creating and managing the network graph, Matplotlib (matplotlib.pyplot) [129] for visualizing the network topology, The Element Tree XML API (xml.etree.ElementTree) [130] for generating and manipulating XML data, and Minimal DOM implementation (xml.dom.minidom) [131] for pretty-printing the XML output. These libraries provide the tools needed to build, configure, and save the network instances in a structured and user-friendly format.

The script accepts user-defined network parameters including bridge and end-system counts, link speeds, TT and BE stream periods, Hyperperiod, application and stream counts, and output XML filename. This allows for generation of customized network instances. Network topology is modeled using NetworkX, representing bridges (BR) and end systems (ES) as nodes. End systems connect randomly to 2-4 bridges, and bridges interconnect similarly, creating a balanced, neither sparse nor dense, network graph. Separate bridge and end-system graphs are merged to form the complete network representation.

Network configuration is saved in XML, defining '<device>' elements (bridges and end systems, including bridge queue types [TT, BE, Nil] and gate control schedules) and '<link>' elements (specifying source, destination, and speed). Links are added in a structured manner for readability and ease of processing. Applications, with associated tasks and streams, generate realistic network workloads. Each application

(unique name, period, type [TT or BE]) includes tasks (name, node, wcet (worst-case execution time), period) randomly assigned to end systems, and streams (name, source, destination, sender/receiver tasks, size, period, rl (redundancy level), priority [TT or BE]). TT streams receive higher priority and gate control schedules aligned with the Hyperperiod, ensuring appropriate handling of time-critical data. Stream assignments balance resource utilization across tasks. The resulting XML file is structured with network description (devices and links), and application details (tasks and streams), enabling comprehensive evaluation and further analysis.

In summary, the problem instances creation process involves collecting user inputs, generating a network topology, configuring devices and links in an XML file, creating applications with tasks and streams, and saving the configuration for further use. The script ensures that the generated instances are realistic, diverse, and suitable for testing and evaluating network management strategies. By automating this process, the script simplifies the creation of complex network scenarios, enabling researchers and engineers to focus on analyzing and optimizing network performance.

3.9.2 OHDSR Problem Instances

The instances of the OHDSR framework were categorized into four groups based on the number of bridges present: 6, 12, 18, and 24. Each group encompasses six distinct problem instances, further divided into two subgroups based on the ratio of end-systems to bridges. The first subgroup maintains a 1.5:1 ratio, representing networks with a moderate density of end-systems relative to bridges. In contrast, the second subgroup employs a 2:1 ratio, simulating networks with a higher concentration of end-systems. This variation in network configurations, as detailed in Table 4.1, allows us

to assess the framework's adaptability and performance across a spectrum of realistic network scenarios.

The experimental setup utilizes bi-directional, full-duplex $1Gbps$ links connecting end-systems to bridges and bridges to each other. Network topologies are generated using a custom NetworkX algorithm [127], randomly interconnecting bridges and connecting each end-system to 2-4 randomly selected bridges, preventing direct end-system-to-end-system links. To evaluate framework performance across diverse application requirements, three subgroups with varying stream configurations are used: (1) predominantly TT streams; (2) equal TT and BE streams; and (3) predominantly BE streams. Approximately 1 out of every 4 streams within each application are designated as multicast to enhance scenario realism. This methodology ensures comprehensive assessment under varying traffic patterns and priorities while maintaining consistent network topology.

To enhance realism, application task counts vary (from 43 to 224, scaling with stream count), with randomly assigned worst-case execution times equal to three percent of their respective task periods. Stream sizes are randomly selected below the 1500-byte MTU, accounting for frame overhead. This variation in task and stream attributes allows for evaluation under diverse workload and data transmission conditions. A $1000\mu s$ application period establishes a consistent temporal framework. Each bridge utilizes eight queues: two for TT streams, two for BE streams, and four reserved for future expansion.

Table 3.6 : Problem Instances for the OHDSR Model

Instance Name	No. BRs	No. ESs	No. Tasks	No. TT Streams	No. BE Streams
Exemplary	4	4	7	2	1
6-9-A	6	9	43	6	12
6-9-B	6	9	43	9	9
6-9-C	6	9	45	12	6
6-12-A	6	12	56	8	16
6-12-B	6	12	57	12	12
6-12-C	6	12	56	16	8
12-18-A	12	18	84	12	24
12-18-B	12	18	87	18	18
12-18-C	12	18	84	24	12
12-24-A	12	24	116	16	32
12-24-B	12	24	112	24	24
12-24-C	12	24	113	32	16
18-27-A	18	27	128	18	36
18-27-B	18	27	129	27	27
18-27-C	18	27	130	36	18
18-36-A	18	36	169	24	48
18-36-B	18	36	165	36	36
18-36-C	18	36	172	48	24
24-36-A	24	36	173	24	48
24-36-B	24	36	171	36	36
24-36-C	24	36	176	48	24
24-48-A	24	48	227	32	64
24-48-B	24	48	223	48	48
24-48-C	24	48	224	64	32

Bridge queues operate under a prioritized gating schedule managed by a Gate Control List (GCL) synchronized to the application Hyperperiod. Time-Triggered (TT) queue gates are prioritized, opening for an initial duration ranging from 50% to 60% of the Hyperperiod. This is followed by a short closed period, between 0% and 5% of the Hyperperiod, to ensure clean transitions between traffic types. Subsequently, Best-Effort (BE) queue gates open for the remainder of the Hyperperiod. The gating time for the OHDSR problem instances are as follows: At $t_1 = 0\mu s$ (GCL: 00000011), at $t_2 = 500\mu s$ (GCL: 00000000), at $t_3 = 550\mu s$ (GCL: 11000000), and at $t_4 = 1000\mu s$ (GCL: 00000000). This mechanism of prioritized gating is designed to optimize resource utilization on the bridges and effectively handle the distinct timing

requirements of both TT and BE streams. The problem instances generated based on these criteria, used to evaluate the OHDSR framework, are presented in Table 3.6.

3.9.3 OHDSR+ Problem Instances

The OHDSR+ instances, inspired by real-world network topologies, are shown in Table 3.7. They vary in bridge-to-end-system ratios. Half of the instances maintain equal numbers of bridges and end-systems, while the other half have twice as many end-systems. This allows for the analysis of framework performance under different network configurations. As in the OHDSR framework, eight problem instances with realistic network topologies were generated. Direct end-system-to-end-system links were prohibited, reflecting typical architectures where end-user devices communicate via intermediary infrastructure. Each bridge and end-system had at least two randomly assigned links to other bridges or end-systems, respectively, enhancing redundancy. The maximum number of links per end-system was four, balancing connectivity and complexity (each link connecting to a distinct node). All links were full-duplex, $1\,Gbps$ connections, ensuring ample bandwidth for evaluation under realistic conditions. A 2:1 TT-to-BE stream ratio was used, with 25% of streams designated as multicast. Redundancy (40%, 70%, or 100%) was applied only to TT streams. Each application handled either only TT or only BE streams (minimum four streams per application), resulting in 33-174 tasks per instance.

Table 3.7 : Problem Instances for the OHDSR+ Model

Instance Name	No. BRs	No. ESs	No. Tasks	No. TT Streams	No. BE Streams
Exemplary	4	4	7	2	1
06B06E	6	6	33	10	5
06B12E	6	12	42	12	6
12B12E	12	12	65	18	9
12B24E	12	24	98	28	14
18B18E	18	18	114	32	16
18B36E	18	36	130	36	18
24B24E	24	24	147	42	21
24B48E	24	48	174	48	24

The tasks in the network are distributed among TT and BE streams based on the type of application they belong to. For example, if there are 10 TT streams and 2 TT applications, each TT application will have approximately 5 streams, with one application potentially having an extra stream if the total is not evenly divisible. The number of tasks for each TT application is calculated as roughly twice the number of streams, plus some additional overhead. For instance, if an application has 5 streams, it might have 12 tasks (calculated as $(5 * 2) + ceil(5 * 0.2)$). Similarly, for BE applications, if there are 8 BE streams and 2 BE applications, each application will have around 4 streams, with one application possibly having an extra stream. The number of tasks for BE applications is determined in the same way as TT applications, ensuring sufficient tasks to handle the streams. This distribution ensures that TT streams, which are time-sensitive, have enough tasks to manage their requirements, while BE streams, which are less critical, are allocated tasks and streams proportionally.

The maximum number of links per end-system was four, balancing connectivity and complexity (each link connecting to a distinct node). All links were full-duplex,

$1\,Gbps$ connections, ensuring ample bandwidth for evaluation under realistic conditions. A 2:1 TT-to-BE stream ratio was used, with 25% of streams designated as multicast. Redundancy (40%, 70%, or 100%) was applied only to TT streams. Each application handled either only TT or only BE streams (minimum four streams per application), resulting in 33–174 tasks per instance.

Task WCETs were capped at a maximum of three percent of their period; stream sizes (including overhead) were limited to $1500\,bytes$ (MTU); all tasks, streams, and applications shared a $1000\mu s$ period. Bridges have eight queues: two for TT traffic (opening at the Hyperperiod start, closing at 45-60%), and two for BE traffic (opening at 60-70%, closing at the Hyperperiod end). The gating time for the OHDSR+ problem instances are as follows: At $t_1 = 0\mu s$ (GCL: 00000011), at $t_2 = 600\mu s$ (GCL: 00000000), at $t_3 = 650\mu s$ (GCL: 11000000), and at $t_4 = 1000\mu s$ (GCL: 00000000). This, combined with realistic network configurations and stream/task parameters, provides a robust evaluation platform for OHDSR+.

3.10 Chapter Summary

This chapter outlines the research methodology used in the study. It begins with a detailed description of the system model, which includes both the network model and the application model. A significant focus is placed on the concept of Queue-Gating Time (QGT) for the bridge, including its modeling. The chapter then presents exemplary problem instances, covering problem formulation, unicast and multicast scenarios, and considerations for redundancy. The latter part of the chapter introduces an optimization framework and provides a comprehensive breakdown of optimization constraints. These constraints cover various aspects such as routing, priority

scheduling, stream scheduling, task scheduling, and reliability considerations for both stream paths and schedules. Finally, the chapter concludes with a discussion on objective functions, which relate to the optimization goals of the study. Overall, this chapter provides a detailed explanation of the research approach, system modeling, and the mathematical framework used to address Time-Sensitive Networking challenges.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

To evaluate the effectiveness and efficiency of the optimization framework, a thorough performance assessment is conducted using a range of synthetic problem instances. These instances are carefully designed to reflect diverse network scenarios and application requirements, ensuring a comprehensive evaluation of the framework's capabilities. The setup employed for these tests is specifically tailored to align with the structure and constraints of our optimization framework. This ensures that the results accurately reflect the framework's performance under realistic conditions and provides valuable insights into its strengths and potential areas for improvement.

4.2 Validation of the Benchmark

This section validates the benchmark work used in comparison with the models presented in Chapter 3. Specifically, the work is compared to REU-CP, as presented in [57]. To the best of our knowledge, REU-CP represents one of the latest frameworks investigated for joint routing and scheduling optimization within Time-Sensitive Networks (TSN). These frameworks demonstrate superior optimization time compared to other CP-based optimization approaches for scheduling and routing in TSN. Furthermore, they achieve notable scalability across different problem instances, varying in the number of bridges (switches) and end systems, while maintaining exceptional latency and worst-case latencies for Time-Triggered (TT) communications.

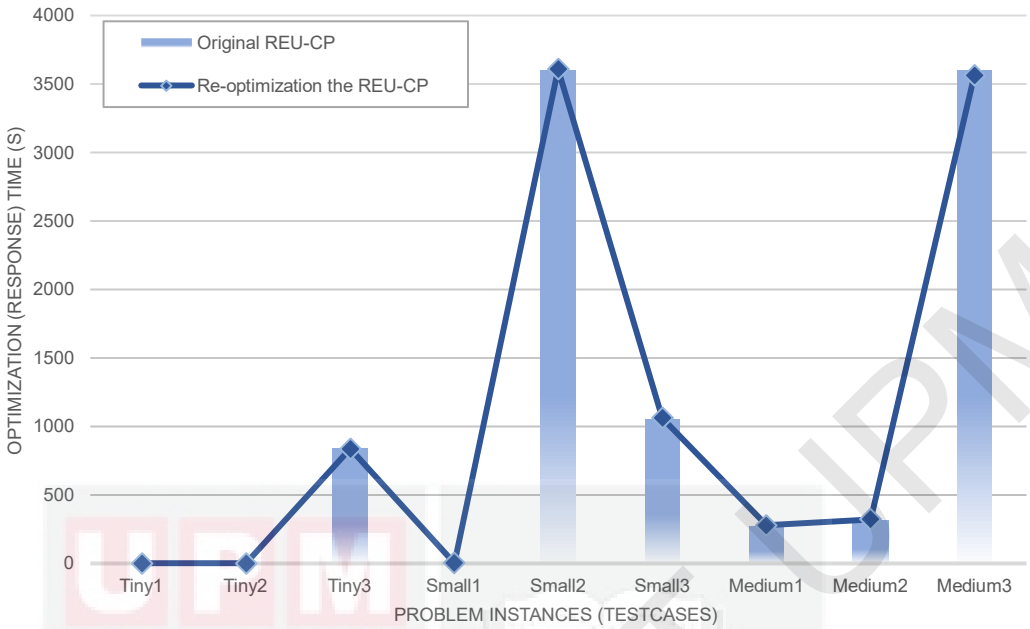

Figure 4.1 : The Total Response Time Comparison between the Original REU-CP Model and the Re-Optimization of the REU-CP Model

As illustrated in Figure 4.1, the re-implementation of REU-CP yielded nearly identical optimization time (response time) and scalability as the original frameworks, with a variance of ±1%. This consistency arises from utilizing the same problem instance scenarios and system design settings employed by the authors in [57] for optimization. Additionally, their work was built using CP-SAT Solver [63], a key component of the OR-Tools suite developed by Google, which is the same toolbox employed in our re-implementation of REU-CP with similar problem instances. This ensures a fair and accurate comparison between our proposed models and the benchmark frameworks.

4.3 Performance Evaluation for Optimized Hybrid Deterministic Scheduling and Routing (OHDSR) framework

The OHDSR optimization framework was evaluated using a rigorous process involving diverse synthetic problem instances designed to test its capabilities and limitations. Analysis of solution quality, computational efficiency, and scalability

under varying conditions identified OHDSR's strengths and weaknesses, informing future refinements and guiding its application to real-world problems. OHDSR effectively minimized total latency across diverse network scenarios (BE-dominant, balanced TT-BE, TT-dominant). While latency increased with network scale, as expected, the relative proportion of traffic types minimally affected overall latency, demonstrating OHDSR's adaptability to varying traffic patterns.

Analysis shows OHDSR prioritizes TT traffic, ensuring timely delivery even under congestion. While BE streams experienced higher worst-case latency than TT streams in BE-dominant scenarios, BE latency remained acceptable in TT-dominant scenarios. This balanced approach ensures reliable delivery for all stream types within the TSN environment. Furthermore, OHDSR exhibits significantly faster response times than REU-CP [57] across all tested scenarios, demonstrating improved efficiency and scalability in handling routing and scheduling. This translates to a substantial reduction in total response time.

OHDSR demonstrates superior scheduling efficiency, achieving substantially faster scheduling response times than REU-CP across all scenarios. Additionally, OHDSR exhibits better scalability than REU-CP, consistently delivering optimal solutions even in large-scale networks (such as 24-48-B and 24-48-C) where REU-CP struggles. This demonstrates OHDSR's effectiveness in complex network environments.

4.3.1 Latency Evaluation

This subsection delves into a comprehensive analysis of the latency results obtained for each problem instance. Our primary focus is on evaluating the overall transmission

latency, which encompasses the time elapsed from the initiation of the sender task to the completion of the final receiver task. This metric provides a holistic view of the time required for data to traverse the network and be processed by the intended recipients. Additionally, the worst-case latency experienced by both Time-Triggered (TT) and Best-Effort (BE) streams is examined. This analysis ensures that even under the most challenging conditions, the latency remains within acceptable bounds and does not adversely impact the reception times of these streams. By scrutinizing both overall and worst-case latencies, valuable insights into the efficiency and reliability of the optimization framework across diverse network scenarios and application requirements are gained.

The gating time configurations for the OHDSR problem instances are defined at specific time intervals, each associated with a unique Gate Control List (GCL). At $t_1 = 0\mu s$, the GCL is set to 00000011, indicating the initial state of the gating mechanism. At $t_2 = 500\mu s$, the GCL transitions to 00000000, effectively closing the gates. This is followed by another transition at $t_3 = 550\mu s$, where the GCL changes to 11000000, reopening specific gates. Finally, at $t_4 = 1000\mu s$, the GCL reverts to 00000000, closing the gates once again. These gating time configurations are critical in determining the latency characteristics of the network, as they influence the scheduling and transmission of both TT and BE streams.

Table 4.1 : The Total Latency, the Worst-Case Latency for Time-Triggered (TT) Streams, and the Worst-Case Latency for Best Effort (BE) Streams, for the BE-Dominant Problem Instances

Instance Name	Total Latency (μs)	Worst-Case Latency of TT streams (μs)	Base Latency for BE streams (μs)	Worst-Case Latency of BE streams (μs)	Total Worst-Case Latency of BE streams (μs)
6-9-A	917	57	550	61	611+
6-12-A	1313	63	550	62	612+
12-18-A	1910	56	550	64	614+
12-24-A	2553	62	550	62	612+
18-27-A	2896	65	550	62	612+
18-36-A	3862	64	550	64	614+
24-36-A	3942	62	550	64	614+
24-48-A	5172	64	550	65	615+

Table 4.1 presents the total latency, worst-case latency for TT streams, and worst-case latency for BE streams across various BE-dominant problem instances. The total latency ranges from $917\mu s$ in the 6-9-A instance to $5172\mu s$ in the 24-48-A instance, reflecting the increasing complexity and scale of the network. The worst-case latency for TT streams remains relatively stable, varying between $56\mu s$ and $65\mu s$ across all instances. In contrast, the base latency for BE streams is consistently $550\mu s$, with the worst-case latency for BE streams ranging from $61\mu s$ to $65\mu s$. When combined, the total worst-case latency for BE streams ranges from $611\mu s$ to $615\mu s$, indicating a minimal increase over the base latency. These results highlight the efficiency of the gating mechanism in managing latency, particularly for TT streams, while also demonstrating the predictable behavior of BE streams under the given constraints.

Network complexity refers to the level of intricacy and interconnectedness, as well as the challenges in managing or analyzing it. This complexity is influenced by factors such as the number of devices in the network, including bridges and end-systems, the

number of tasks performed at the end-systems, and the variety and volume of traffic. Different traffic types, such as TT and BE, add to the challenge with their unique requirements. These elements combine to create a dynamic environment that requires careful planning and optimization for efficient and reliable performance.

Table 4.2 : The Total Latency, the Worst-Case Latency for Time-Triggered (TT) Streams, and the Worst-Case Latency for Best Effort (BE) Streams, for the Balanced TT-BE Problem Instances

Instance Name	Total Latency (µs)	Worst-Case Latency of TT streams (µs)	Base Latency for BE streams (µs)	Worst-Case Latency of BE streams (µs)	Total Worst-Case Latency of BE streams (µs)
6-9-B	948	60	550	60	610+
6-12-B	1277	59	550	62	612+
12-18-B	1942	65	550	60	610+
12-24-B	2538	59	550	63	613+
18-27-B	2944	64	550	62	612+
18-36-B	3927	68	550	62	612+
24-36-B	3929	66	550	62	612+
24-48-B	5270	65	550	66	616+

Table 4.2 provides a detailed analysis of the total latency and worst-case latency for TT and BE streams across balanced TT-BE problem instances. The total latency values exhibit a progressive increase, ranging from $948\mu s$ in the 6-9-B instance to $5270\mu s$ in the 24-48-B instance, reflecting the growing complexity and scale of the network configurations. The worst-case latency for TT streams remains relatively consistent, varying between $59\mu s$ and $68\mu s$, with the highest value observed in the 18-36-B instance. This stability underscores the predictable behavior of TT streams, which are designed to meet stringent timing requirements.

While the worst-case latency for BE streams ranges from $60\mu s$ to $66\mu s$, when combined with the base latency (representing the open time of the BE gates), the total

worst-case latency for BE streams ranges from $610\mu s$ to $616\mu s$, indicating a marginal increase compared to the base latency. This minimal variation suggests that the balanced TT-BE scheduling mechanism effectively manages the coexistence of TT and BE streams, ensuring that BE traffic does not significantly disrupt the deterministic performance of TT streams. The results highlight the robustness of the scheduling approach in maintaining low and predictable latencies for both stream types, even as the network scales in size and complexity. This balance is critical for applications requiring both real-time guarantees and efficient handling of non-critical traffic.

Table 4.3 examines the total latency and worst-case latency for TT and BE streams across TT-dominant problem instances. The total latency increases progressively, ranging from $96\mu s$ in the 6-9-C instance to $5220\mu s$ in the 24-48-C instance, reflecting the growing network complexity. The worst-case latency for TT streams remains stable, varying between $58\mu s$ and $68\mu s$, with the highest value observed in the 18-27-C instance. This consistency highlights the deterministic nature of TT streams, which are prioritized in TT-dominant configurations.

Table 4.3 : The Total Latency, the Worst-Case Latency for Time-Triggered (TT) Streams, and the Worst-Case Latency for Best Effort (BE) Streams, for the TT-Dominant Problem Instances

Instance Name	Total Latency (µs)	Worst-Case Latency of TT streams (µs)	Base Latency for BE streams (µs)	Worst-Case Latency of BE streams (µs)	Total Worst-Case Latency of BE streams (µs)
6-9-C	966	60	550	58	608+
6-12-C	1228	58	550	59	609+
12-18-C	1926	60	550	63	613+
12-24-C	2638	63	550	60	610+
18-27-C	2908	68	550	60	610+
18-36-C	3915	64	550	62	612+
24-36-C	3928	63	550	65	615+
24-48-C	5220	66	550	62	612+

The base latency for BE streams is consistently $550\mu s$, while the worst-case latency for BE streams ranges from $58\mu s$ to $65\mu s$. When combined, the total worst-case latency for BE streams ranges from $608\mu s$ to $615\mu s$, showing only a slight increase over the base latency. This slight change highlights the TT-dominant scheduling mechanism's ability to effectively prioritize time-sensitive (TT) streams while ensuring acceptable latency for best-effort (BE) traffic. The results underscore the efficiency of the approach in ensuring real-time performance for critical TT streams, even as the network scales, without significantly compromising the handling of BE traffic.

Figure 4.2 provides a visual representation of the distribution of total latencies across various network scales and stream dominance scenarios. As previously established, larger networks with a higher number of streams tend to exhibit greater total latency due to increased competition for network resources and scheduling complexities.

However, an interesting observation emerges: the dominance of one stream type over another appears to have a minimal impact on the overall latency.

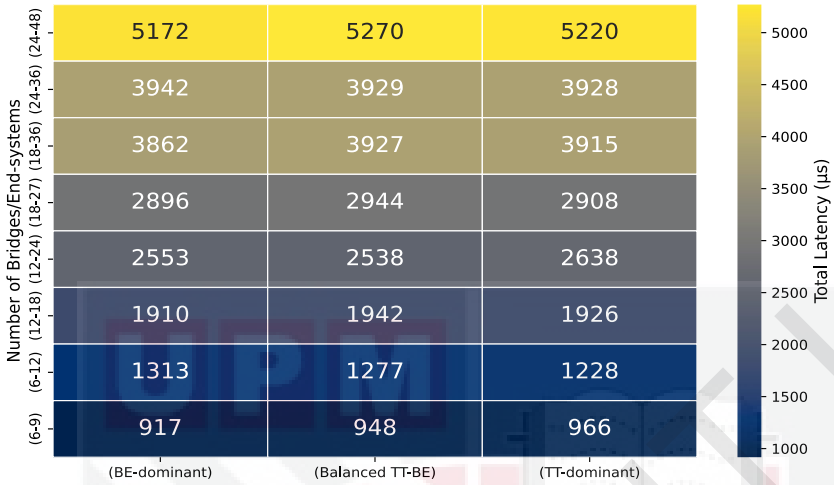

Figure 4.2 : Total Latency Comparison across Three Scenarios: BE-Dominant, Balanced TT-BE, and TT-Dominant Scenarios, for Various Scale Instances

This suggests that our model effectively adapts to different traffic patterns and maintains efficient performance regardless of the relative proportions of TT and BE streams. The most notable difference in total latency for networks of similar scale but varying stream dominance is observed in the 12-24 scale scenarios. Here, a discrepancy of $100\mu s$ exists between the balanced TT-BE and TT-dominant scenarios, representing a marginal increase of 3.94%. Crucially, this slight increase in latency does not compromise the overall performance of the network, demonstrating the robustness and adaptability of our optimization framework across diverse operating conditions.

Overall, across all configurations (BE-dominant, balanced TT-BE, and TT-dominant problem instances), the total latency increases with increasing network complexity. However, the worst-case latency for TT streams remains stable, demonstrating the

robustness of Time-Triggered scheduling in meeting real-time requirements. The base latency for BE streams is consistently $550\mu s$, with the total worst-case latency for BE streams showing only marginal increases, ranging from $608\mu s$ to $616\mu s$. This indicates that the scheduling mechanisms effectively balance the demands of TT and BE streams, ensuring deterministic performance for critical traffic while maintaining reasonable latencies for non-critical traffic. These findings highlight the scalability and efficiency of the OHDSR framework in diverse network scenarios, making it suitable for applications requiring both real-time guarantees and flexible handling of best-effort communication.

4.3.2 Response Time Evaluation

Beyond latency, the response time of the model solver, which represents the duration it takes to identify a feasible solution to the optimization problem, is also assessed. Solver response time was analyzed, encompassing total, scheduling, and routing components, across BE-dominant, balanced TT-BE, and TT-dominant scenarios and network scales, to assess computational efficiency and real-time performance. Response time, like latency, increases with network size. However, within the same network scale (e.g., 18-36 node instances), variations across different traffic mixes were minimal (maximum difference of $3670ms$ or 8.75% between TT-dominant and BE-dominant scenarios), indicating that the relative proportion of TT and BE streams has a limited effect on solution time.

Scheduling response time consistently remained lower than routing response time across all scenarios. While minimal, slightly higher scheduling response times were observed in larger-scale TT-dominant scenarios, due to increased complexity in

prioritizing numerous time-sensitive streams. Response time increased with network scale, exhibiting a more pronounced rise at the 24-48 node scale, particularly in TT-dominant scenarios.

To gain a deeper understanding of the findings and assess the relative performance of the framework, the response time results are compared with previous research in the field. Specifically, the results are benchmarked against the work presented in [57] (REU-CP), which tackled a similar joint scheduling and routing problem using a constraint programming (CP) model. The authors of REU-CP provide open access to their model, enabling us to conduct a direct comparison using the same problem instances from Table 3.6 across various scenarios. It's worth noting that REU-CP uses the term "optimization time" instead of "response time," but the underlying concept remains the same. For clarity and ease of comparison, the benchmark results are presented across the three distinct scenarios: BE-dominant, balanced TT-BE, and TT-dominant. This allows us to assess the relative performance of our framework under different network conditions and traffic patterns.

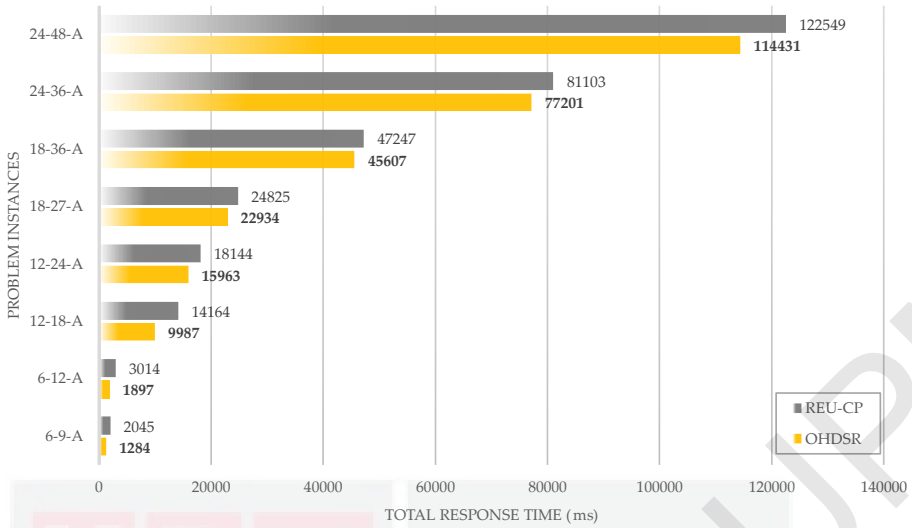

Figure 4.3 : The Total Response Time Comparison between Our Proposed OHDSR Model and the Related Work REU-CP [57] **Model for the BE-Dominant Problem Instances**

Figure 4.3 illustrates a comparative analysis of the total response times between the proposed OHDSR model and the related REU-CP model [57] for BE-dominant problem instances. The figure employs a bar chart to depict the response times across various problem instances, with the x-axis representing the problem instances and the y-axis quantifying the total response time in milliseconds (ms). The OHDSR model consistently demonstrates superior performance, with response times significantly lower than those of the REU-CP model. For instance, in the smallest problem instance, the OHDSR model achieves a response time of $1284ms$ compared to $2045ms$ for REU-CP. This trend continues across larger instances, with OHDSR recording $9987ms$ versus $14164ms$, $22934ms$ versus $24825ms$, and $114431ms$ versus $122549ms$ in the largest instance. These results highlight the efficiency of the OHDSR model, which reduces response times by approximately 37.21% to 6.63% across the range of instances. The figure also includes numerical annotations above each bar, providing precise response time values for both models. This comparative

visualization underscores the effectiveness of the OHDSR model in optimizing resource utilization and scheduling strategies, making it a more efficient solution for BE-dominant problem instances.

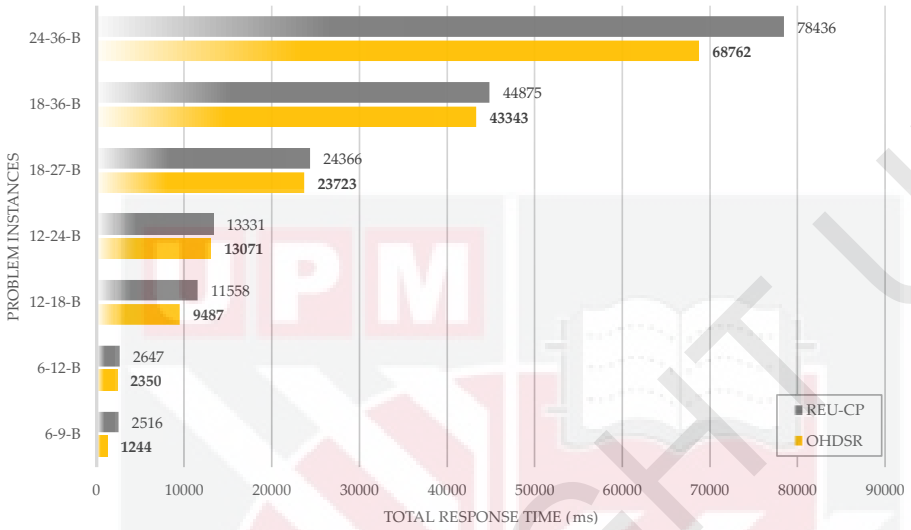

Figure 4.4 : The Total Response Time Comparison between Our Proposed OHDSR Model and the Related Work REU-CP [57] **Model for the Balanced TT-BE Problem Instances**

Figure 4.4 provides a detailed comparison of the total response times between the proposed OHDSR model and the REU-CP model [57] for balanced TT-BE problem instances, emphasizing the performance superiority of OHDSR. The data reveals that OHDSR consistently outperforms REU-CP across all problem instances, with response times significantly lower in every case. For instance, in the smallest instance, OHDSR achieves a response time of $1244ms$, which is nearly half of the $2516ms$ recorded by REU-CP. This substantial difference persists as the problem size increases, with OHDSR recording $9487ms$ versus $11558ms$, $23723ms$ versus $24366ms$, and $68762ms$ versus $78436ms$ in the largest instance.

The results for the balanced TT-BE problem instances indicate that OHDSR not only excels in smaller instances but also maintains its efficiency as the complexity of the problem grows. The consistent reduction in response times, ranging from approximately 50.48% in smaller instances to around 12.35% in larger ones, highlights the robustness and scalability of the OHDSR model. This performance advantage can be attributed to OHDSR's optimized resource allocation and scheduling strategies, which effectively minimize delays and enhance overall system responsiveness. The quantitative evidence underscores OHDSR's potential as a more efficient and reliable solution for handling balanced TT-BE problem instances compared to the REU-CP model.

Figure 4.5 provides a comparative analysis of TT-dominant problem instances, demonstrating that the OHDSR model achieves performance levels comparable to those observed in both balanced TT-BE and TT-dominant scenarios. For example, in the smallest instance, OHDSR records a response time of $1512ms$, compared to $1946ms$ for REU-CP. This trend continues as the problem size increases, with OHDSR achieving $8783ms$ versus $9555ms$, $24319ms$ versus $24940ms$, and $65540ms$ versus $75281ms$ in the largest instance.

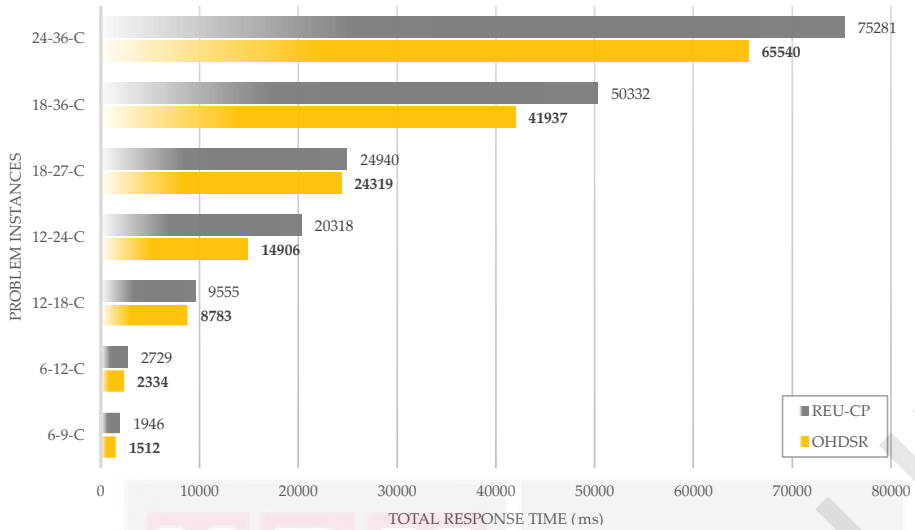

Figure 4.5 : The Total Response Time Comparison between Our Proposed OHDSR Model and the Related Work REU-CP [57] **Model for the TT-Dominant Problem Instances**

These results demonstrate that the OHDSR model effectively reduces response times, achieving reductions of approximately 22.3% in smaller instances, 12.93% in larger instances, and up to 26.36% for the 12-24-C problem instance compared to the REU-CP model. The quantitative data underscores the efficiency of the OHDSR model in handling TT-dominant problem instances, demonstrating its ability to optimize resource allocation and scheduling strategies effectively. This consistent performance improvement across varying problem sizes further validates OHDSR as a robust and scalable solution compared to the REU-CP model.

TSN hinges upon efficient scheduling mechanisms to guarantee timely and reliable delivery of time-sensitive data. To thoroughly assess the effectiveness of the proposed scheduling model, a comparative analysis of its response time performance against the REU-CP [57] model is undertaken. This evaluation will specifically focus on the same set of scenarios previously employed during the assessment of total response time. By directly contrasting the responsiveness of both models under identical conditions,

deeper insights into their respective strengths and limitations are aimed to be gained, ultimately determining which approach offers superior performance for real-time applications within TSN environments.

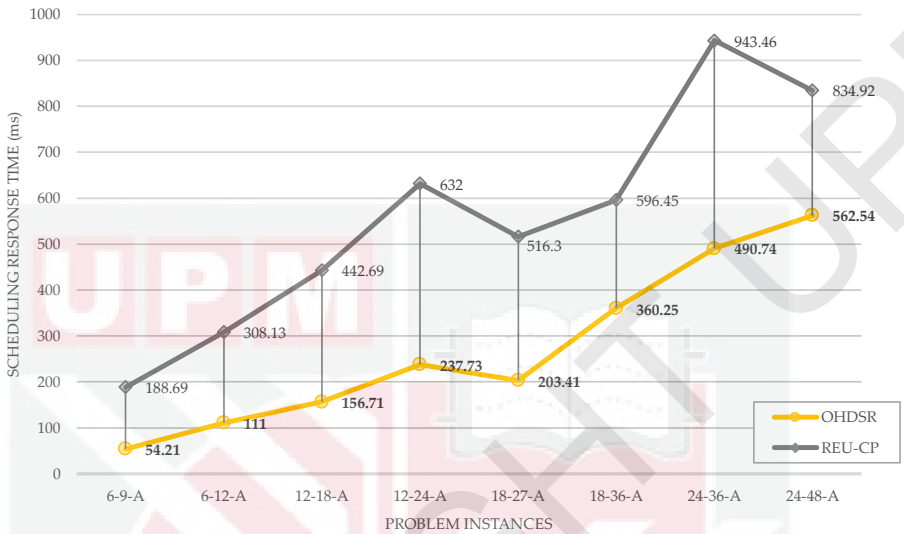

Figure 4.6 : The Scheduling Response Time Comparison between Our Proposed OHDSR Model and the Related Work REU-CP [57] **Model for the BE-Dominant Problem Instances**

While the OHDSR model already demonstrated advantages in total response time, its performance improvement becomes even more pronounced when focusing specifically on scheduling response time. Figure 4.6 vividly illustrates this, showcasing a remarkable reduction of $272.38ms$ (equivalent to 32.62%) for the largest-scale instance. This efficiency gain further escalates to an impressive 71.26% for the smallest-scale instance, highlighting the model's ability to expedite scheduling decisions across diverse network sizes. Furthermore, Figure 4.7 reveals that this trend of superior performance extends to scenarios with a balanced mix of TT and BE streams. With the exception of the 24-48-B instance (where REU-CP failed to find a solution), our model consistently achieves improvements ranging from 28.21% to 65.13% compared to REU-CP. This significant reduction in scheduling response time

translates to faster decision-making and ultimately, a more agile and responsive network environment, particularly beneficial for time-sensitive applications.

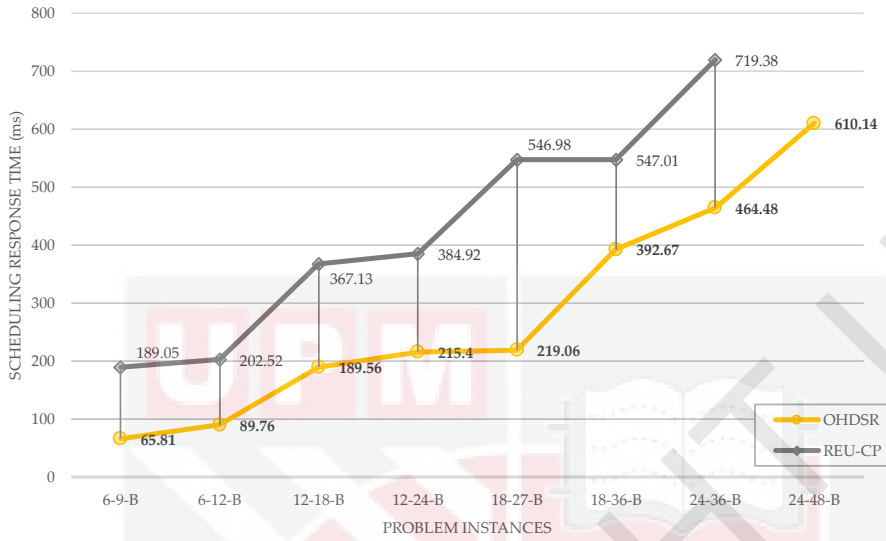

Figure 4.7 : The Scheduling Response Time Comparison between Our Proposed OHDSR Model and the Related Work REU-CP [57] **Model for the Balanced TT-BE Problem Instances**

The third group of instances, depicted in Figure 4.8, characterized by the dominance of TT streams over BE streams, presents a compelling case study for the OHDSR model's scheduling efficiency. Compared to the REU-CP model, OHDSR demonstrates consistently impressive performance, particularly as the instance scale increases. While the largest instance, 24-48-C, lacks a solution in the REU-CP model for direct comparison, the 24-36-C instance offers valuable insights. Here, the OHDSR model achieves a remarkable 46.86% reduction in scheduling response time, a substantial improvement considering the instance size. This result significantly surpasses the gains observed in the other two instance groups (32.62% and 35.44%), further emphasizing the effectiveness of OHDSR in handling complex scheduling scenarios. These findings, along with the previously discussed total response time

improvements, solidify the OHDSR model's superior performance compared to REU-CP across various instance types and sizes.

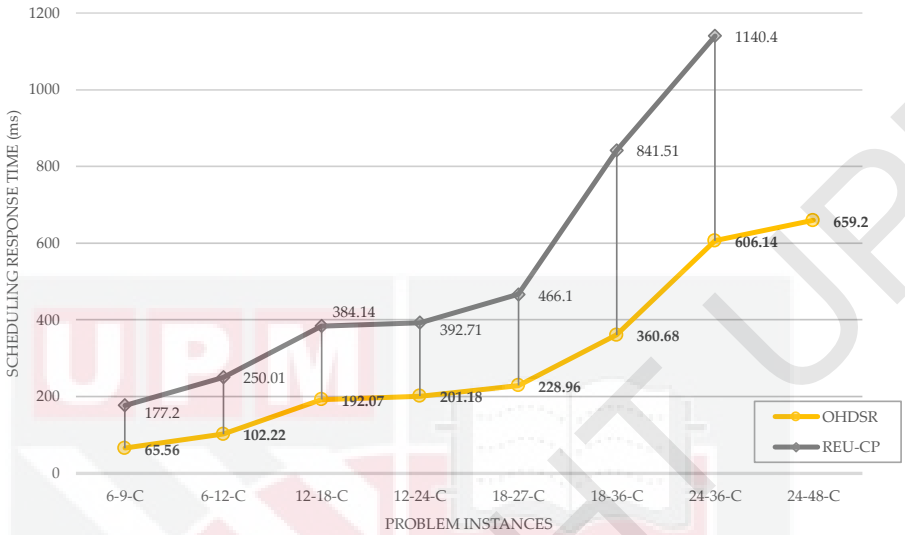

Figure 4.8 : The Scheduling Response Time Comparison between Our Proposed OHDSR Model and the Related Work REU-CP [57] **Model for the TT-Dominant Problem Instances**

The increased complexity of larger networks, such as the 24-48-B and 24-48-C instances, led to a notable rise in total response time compared to smaller-scale networks. Despite this challenge, the OHDSR model consistently delivered optimal solutions while maintaining a stable response time for the scheduling model. In contrast, when these same instances were executed using the REU-CP model, the cp-sat solver failed to identify any solution, returning an "UNKNOWN" value. This difference in performance, as evidenced by the consistent results across various instances run on the same system (Table 4.4), clearly demonstrates the superior scalability of OHDSR compared to REU-CP. While REU-CP struggles to handle the complexity of larger networks, OHDSR provides efficient and effective solutions, solidifying its position as a more robust and scalable approach for network resource management.

Table 4.4 : Comparison between Our Proposed OHDSR Model and the Related Work REU-CP Model for the Total Response Time and Scheduling Response Time for the 24-48 Scale Problem Instances

Instance Name	OHDSR		REU-CP	
	Total Response Time (ms)	Scheduling Response Time (ms)	Total Response Time (ms)	Scheduling Response Time (ms)
24-48-A	114431	562.54	122549	834.92
24-48-B	522403	610.14	/	/
24-48-C	581578	659.20	/	/

The routing response time is a critical metric for evaluating the scalability and efficiency of the routing mechanism. The routing response times for the BE-dominant (A), balanced TT-BE (B), and TT-dominant (C) problem instances are presented in Table 4.5, demonstrating the computational overhead associated with the OHDSR model across varying network configurations.

Table 4.5 : The Routing Response Time for the BE-Dominant, Balanced TT-BE, and TT-Dominant Problem Instances

Instance Name	Routing Response Time (ms) - OHDSR		
	BE-dominant (A)	balanced TT-BE (B)	TT-dominant (C)
6-9	1080.79	1014.60	1245.03
6-12	1610.06	1999.34	1997.99
12-18	8833.36	8172.70	7832.07
12-24	14143.91	10999.18	13484.07
18-27	19623.62	20947.31	21725.65
18-36	41851.81	38690.48	37209.78
24-36	71890.85	61823.00	59632.47
24-48	104935.49	513068.57	573310.41

For smaller instances, such as 6-9 and 6-12, the routing response times are relatively low, ranging from $1014.60ms$ to $1999.34ms$, with the BE-dominant configuration generally exhibiting slightly faster response times compared to the balanced and TT-dominant configurations. However, as the network scales to larger instances, such as

12-18 and beyond, the routing response times increase significantly, reaching up to $104935.49ms$ for the 24-48 BE-dominant instance. Notably, the TT-dominant configuration often results in higher response times compared to the BE-dominant and balanced configurations, particularly in larger instances, as seen in the 24-48 instance where the TT-dominant response time ($573310.41ms$) far exceeds that of the BE-dominant ($104935.49ms$) and balanced ($513068.57ms$) configurations. This trend suggests that the prioritization of TT streams in TT-dominant instances introduces additional computational complexity, leading to longer routing response times.

In conclusion, this evaluation demonstrates that the OHDSR model offers significant improvements in both total and scheduling response times compared to the REU-CP model across various network scales and traffic mixes. While response time increases with network size, the impact of traffic mixes on solution time is relatively minor. The OHDSR model consistently outperforms REU-CP, achieving substantial reductions in response time, particularly for larger instances where REU-CP sometimes failed to find a solution. This superior performance, especially evident in the scheduling component, highlights OHDSR's efficiency and scalability in managing network resources, making it a robust and promising solution for real-time applications within Time-Sensitive Networking environments.

4.4 Performance Evaluation for OHDSR+ Framework

The OHDSR+ optimization framework was rigorously evaluated using diverse problem instances to assess its solution quality, computational efficiency (latency and response time), and scalability. This multifaceted evaluation demonstrates OHDSR+'s robustness and versatility in addressing a wide range of optimization challenges. Our

results show that OHDSR+ maintains low latency even as TT redundancy increases, a key requirement for real-time systems. This is achieved by prioritizing the first received stream, whether original or redundant, reducing processing and transmission delays while optimizing network resource use. Furthermore, analysis of worst-case latency (WCL) for TT and BE streams reveals that, despite TT traffic prioritization, WCL values for both remain closely aligned across scenarios. This is due to OHDSR+'s gate control mechanism, which prioritizes TT streams with strict timing constraints while strategically managing gate closures to avoid significant delays for BE traffic. This balance underscores the framework's ability to handle diverse traffic with varying criticality effectively.

To benchmark the performance of OHDSR+, a comparative analysis was conducted against REU-CP [57], a relevant Constraint Programming (CP)-based model. The results demonstrate that OHDSR+ consistently outperforms REU-CP in terms of total latency minimization across all redundancy scenarios and problem instances, highlighting its effectiveness in meeting the stringent timing requirements of real-time applications. OHDSR+ exhibited significantly faster response times than REU-CP across varying network complexities and redundancy levels, a key performance advantage. While both factors influenced response times, their interaction was nuanced. Moreover, OHDSR+'s scheduling model demonstrated superior performance, achieving faster scheduling response times across all instances, contributing to its overall efficiency within the TSN framework.

Finally, scalability, a critical factor in TSN scheduling and routing optimization, was evaluated. OHDSR+ demonstrated superior scalability compared to REU-CP. Tested

across diverse network topologies, stream counts, and tasks, OHDSR+ consistently solved all eight problem instances and their redundancy scenarios, showcasing its adaptability to growing network complexity and workloads. In contrast, REU-CP struggled with larger, more complex networks, failing in specific scenarios. Notably, while REU-CP could not solve the 24B48E instance, OHDSR+ successfully handled it across all redundancy scenarios. This highlights REU-CP's limitations in managing intricate configurations and reinforces OHDSR+ as a robust solution for real-world TSN deployments with varying demands.

4.4.1 Latency Evaluation

The evaluation of any TSN model necessitates the assessment of latency and WCL to ensure compliance with real-time application timing requirements. This section examines both TT and BE transmissions, following the approach outlined in Section 4.3.1. The evaluation provides a comprehensive analysis of OHDSR+'s capability to meet QoS requirements by ensuring low latency and minimizing WCL, thereby enabling predictable and reliable network performance. To thoroughly evaluate the impact of redundancy on latency within the OHDSR+ framework, a series of assessments were conducted using varying levels of redundant TT streams. Starting with a 40% redundancy level across eight diverse problem instances, the redundancy progressively increased to 70% and ultimately to 100%. This systematic approach facilitated an in-depth understanding of the framework's performance under different

Table 4.6 presents the total latency for problem instances executed on the OHDSR+ model under varying levels of redundancy applied to TT streams, specifically 40%, 70%, and 100% redundancy. The results indicate that the total latency remains

relatively stable across all redundancy levels, with only minor variations observed. For instance, in the 06B06E configuration, the total latency is $772\mu s$ for both 40% and 70% redundancy, increasing slightly to $773\mu s$ at 100% redundancy. Similarly, in larger instances such as 24B48E, the total latency fluctuates minimally, ranging from $3919\mu s$ to $3923\mu s$ across the three redundancy levels. This consistency suggests that the OHDSR+ model effectively manages the additional redundancy for TT streams without significantly impacting the overall latency. The stability in total latency across redundancy levels highlights the robustness of the OHDSR+ approach in maintaining efficient network performance, even when higher levels of fault tolerance are introduced. These findings underscore the model's capability to balance redundancy requirements with latency constraints, making it suitable for applications demanding both reliability and real-time performance.

Table 4.6 : The Total Latency for the Problem Instances Run on the OHDSR+ Model, When Applying 40%, 70%, and 100% Redundancy to the Time-Triggered (TT) Streams

Instance Name	Total Latency (μs) - OHDSR+		
	40% TT Redundancy	70% TT Redundancy	100% TT Redundancy
06B06E	772	772	773
06B12E	962	962	962
12B12E	1493	1496	1490
12B24E	2233	2237	2244
18B18E	2621	2633	2635
18B36E	2960	2963	2967
24B24E	3455	3451	3466
24B48E	3923	3922	3919

Table 4.7 presents the worst-case latency for TT streams and BE streams in problem instances executed on the OHDSR+ model with 40% redundancy applied to TT streams. The gating time configuration for these instances is defined at specific

intervals: at $t_1 = 0\mu s$ (GCL: 00000011), at $t_2 = 600\mu s$ (GCL: 00000000), at $t_3 = 650\mu s$ (GCL: 11000000), and at $t_4 = 1000\mu s$ (GCL: 00000000). This gating mechanism ensures controlled access to the network, influencing the latency characteristics of both TT and BE streams. The worst-case latency for TT streams remains consistent across instances, ranging from $55\mu s$ to $65\mu s$, demonstrating the deterministic behavior of TT traffic under the applied redundancy.

Table 4.7 : The Worst-Case Latency (WCL) for Time-Triggered (TT) Streams and the Worst-Case Latency for Best-Effort (BE) Streams in the Problem Instances Run on the OHDSR+ Model, When Applying 40% Redundancy to the TT Streams

Instance Name	Worst-Case Latency of TT streams (µs)	Base Latency for BE streams (µs)	Worst-Case Latency of BE streams (µs)	Total Worst-Case Latency of BE streams (µs)
06B06E	55	650	61	711+
06B12E	61	650	60	710+
12B12E	62	650	61	711+
12B24E	62	650	58	708+
18B18E	62	650	66	716+
18B36E	65	650	62	712+
24B24E	62	650	63	713+
24B48E	61	650	63	713+

For BE streams, the base latency is fixed at $650\mu s$, reflecting the impact of the gating mechanism at t_3. The worst-case latency for BE streams varies slightly, ranging from $58\mu s$ to $66\mu s$. This results in a minimal total worst-case latency for BE streams between $708\mu s$ and $716\mu s$. These results indicate that the OHDSR+ model effectively manages the coexistence of TT and BE streams without significantly compromising the performance of BE traffic. The stability in worst-case latencies for both stream types underscore the robustness of the gating mechanism and the OHDSR+ model in maintaining predictable and efficient network behavior under redundancy constraints.

A comparative analysis of OHDSR+ and REU-CP [57], a Constraint Programming-based model addressing joint scheduling and routing, was conducted using common problem instances. Table 4.8 shows that while REU-CP exhibits acceptable total latency, OHDSR+ consistently achieves lower total latency across all scenarios and instances. The performance improvement of OHDSR+ over REU-CP varies by instance and redundancy level, ranging from a modest 0.06% reduction (40% TT redundancy, 12B12E instance) to a more substantial 1.03% reduction (100% TT redundancy, 06B12E instance).

Table 4.8 : The Total Latency for the Problem Instances Run on the REU-CP Model, When Applying 40%, 70%, and 100% Redundancy to the Time-Triggered (TT) Streams

Instance Name	Total Latency (μs) - REU-CP		
	40% TT Redundancy	70% TT Redundancy	100% TT Redundancy
06B06E	774	774	774
06B12E	966	962	972
12B12E	1494	1498	1502
12B24E	2237	2241	2251
18B18E	2638	2642	2652
18B36E	2971	2976	2984
24B24E	3473	3480	3482
24B48E	/	/	/

Table 4.9 provides a comprehensive overview of the percentage reduction in total latency achieved by OHDSR+ relative to REU-CP across all scenarios and instances. This comparative analysis underscores the consistent superiority of OHDSR+ in minimizing latency, demonstrating its effectiveness in meeting the stringent timing requirements of real-time applications. As shown in Tables 4.8 and 4.9, the 24B48E instance reveals a significant limitation of REU-CP. Marked with "/", REU-CP failed to solve any redundancy scenarios, with its CP-SAT solver returning "UNKNOWN,"

indicating no feasible solution under the instance's constraints. This underscores REU-CP's struggle with larger, more complex topologies, where expanding problem spaces challenge constraint satisfaction. In contrast, OHDSR+ successfully solved the 24B48E instance across all scenarios, demonstrating its robustness and superiority in handling complex real-world applications.

Table 4.9 : The Total Latency Minimization Percentages of the OHDSR+ Model Compared to the REU-CP Model for the Problem Instances, When Applying 40%, 70%, and 100% Redundancy to the Time-Triggered (TT) Streams

Instance Name	Total Latency Minimization Percentages		
	40% TT Redundancy	70% TT Redundancy	100% TT Redundancy
06B06E	0.26%	0.26%	0.13%
06B12E	0.41%	0.41%	1.03%
12B12E	0.06%	0.13%	0.80%
12B24E	0.18%	0.18%	0.31%
18B18E	0.64%	0.34%	0.64%
18B36E	0.37%	0.44%	0.57%
24B24E	0.52%	0.83%	0.46%
24B48E	/	/	/

4.4.2 Response Time Evaluation

In the realm of optimization models, the time it takes to reach an optimal or feasible solution is a crucial performance indicator. This duration, termed "response time," directly impacts the model's practicality and efficiency. It's important to distinguish between two key aspects of response time:

- Total Response Time: This encompasses the entire duration from problem input to solution output, encompassing all internal processes.
- Specific Response Times: These refer to the time taken for individual sub-tasks, such as scheduling or routing.

To avoid ambiguity, the aspect of response time being referred to will be explicitly stated by adding prefixes like "scheduling" or "routing" when necessary. Model response time under varying workloads (40%, 70%, and 100% redundancy in TT streams) was evaluated to assess scalability and performance under diverse conditions. Response time is affected by many factors, such as the complexity of the network, as determined by the number of applications, tasks, streams, and network devices. This complexity has a significant impact on the total response time. In addition to network complexity, the level of redundancy in TT streams also plays a critical role in influencing the total response time.

Figure 4.9 illustrates the total response time for problem instances executed on the OHDSR+ framework, with varying levels of redundancy applied to the Time-Triggered (TT) streams. The figure presents a bar chart where the x-axis represents different problem instances, and the y-axis denotes the total response time in milliseconds (ms). The chart includes three distinct sets of bars, each corresponding to a different redundancy level: 40%, 70%, and 100%. The bar chart uses color-coding to represent different TT stream redundancy levels. Response times for each problem instance are shown, facilitating comparison of redundancy's impact. The x-axis displays problem instances, and the y-axis shows response time (in $100000ms$ increments, up to $1000000ms$).

Quantitatively, the results show a clear trend where higher redundancy levels lead to increased response times. For instance, at 40% redundancy, the response times range from $694ms$ to $555950ms$ across the problem instances. When redundancy is increased to 70%, the response times rise, with values ranging from $813ms$ to

$526974ms$. At 100% redundancy, the response times further increase, ranging from $1006ms$ to $954127ms$.

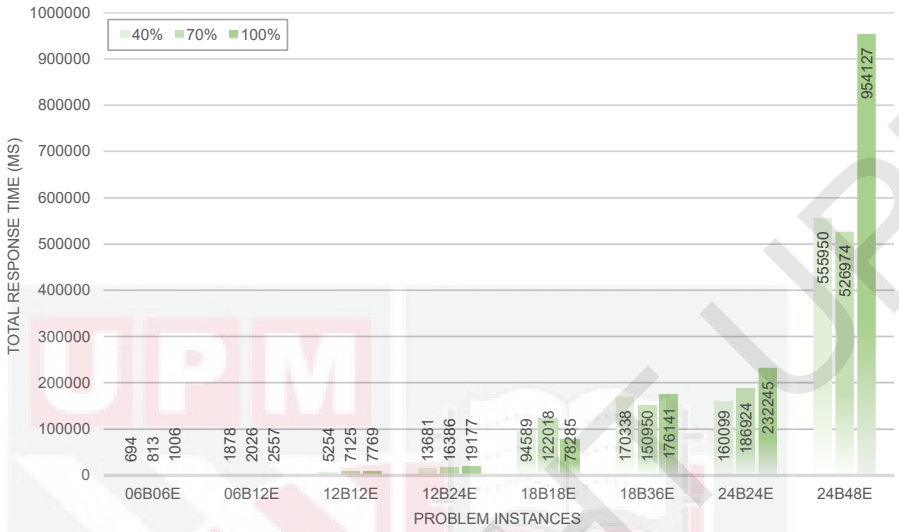

Figure 4.9 : The Total Response Time for the Problem Instances Run on the OHDSR+ Model, When Applying 40%, 70%, and 100% Redundancy to the Time-Triggered (TT) Streams

The observation that the total response time for 40% redundancy is higher than for 70% redundancy in some cases, such as problem instances 18B36E and 24B48E, and that the total response time for 40% and 70% redundancy is higher than for 100% redundancy in other cases, such as problem instance 18B18E, can be attributed to the inherent behavior of constraint programming (CP) and the CP-SAT solver [63] from Google OR-Tools.

In certain instances, a higher redundancy level might inadvertently lead to a simplification of the problem, narrowing down the solution space and making it easier for the model to identify the optimal solution. Additionally, the search space itself can be influenced by redundancy levels, potentially making it more conducive to efficient exploration by the model under certain scenarios. Furthermore, the random nature of

network connections in our instances adds another layer of complexity. The specific arrangement of connections, when coupled with a particular redundancy level, might serendipitously create a scenario that is easier for the model to solve compared to other configurations with lower redundancy.

This complex interplay of factors underscores the challenges in predicting response times and emphasizes the need for a detailed analysis of individual problem instances to fully understand and optimize model performance. These factors collectively explain why, in some cases, a 40% redundancy level results in higher response times than 70%, highlighting the nuanced trade-offs in constraint-based optimization and the importance of carefully calibrating redundancy levels to achieve optimal performance.

To gauge the performance of the model, a comparative analysis was conducted with the REU-CP model [57]. The OHDSR+ model demonstrated superior performance compared to the REU-CP model across all problem instances and redundancy levels. Notably, the REU-CP model failed to solve the 24B48E instance (refer to Section 4.4.3). Let us focus on the 40% TT redundancy scenario, with results presented in Figure 4.10. The data clearly demonstrates that OHDSR+ consistently achieved shorter response times than the REU-CP model across all evaluated instances. The most significant improvement was observed in the 12B12E instance, where OHDSR+ achieved a 19.66% reduction in response time. Even the smallest improvement, observed in the 12B24E instance, resulted in a noteworthy 6.80% reduction, translating to a $998ms$ faster processing time for OHDSR+. These results clearly

showcase the efficiency and effectiveness of our proposed model in comparison to existing approaches.

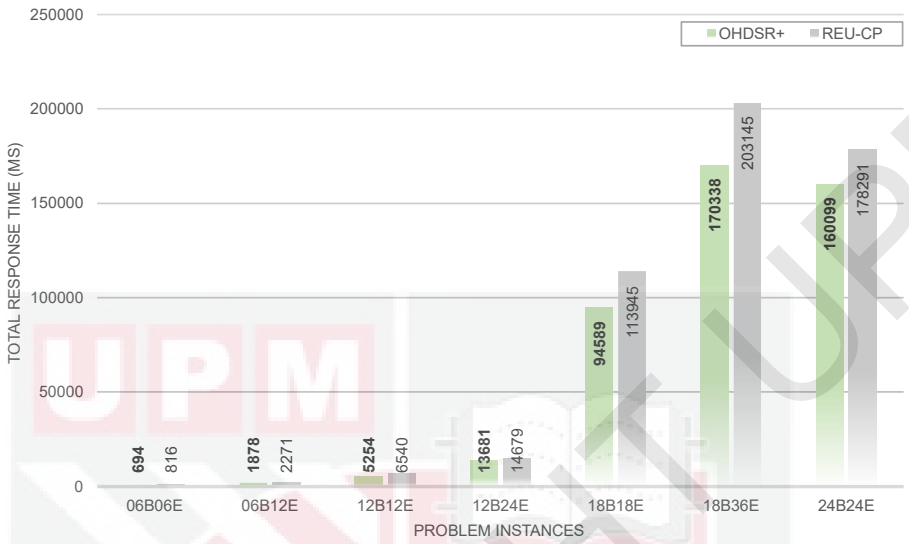

Figure 4.10 : The Total Response Time Comparison between Our Proposed OHDSR+ Model and the Related Work REU-CP [57] **Model When Applying 40% Redundancy to the Time-Triggered (TT) Streams**

Moving on to the 70% TT redundancy scenario, Figure 4.11 provides a comparative analysis of the total response times for both OHDSR+ and REU-CP. The results further solidify the superior performance of our proposed model. In most instances, OHDSR+ exhibits a significant advantage over the REU-CP model. A particularly noteworthy observation is seen in the 06B06E instance, where OHDSR+ achieves an impressive 34.70% reduction in total response time compared to REU-CP. This marks the most substantial improvement observed in this scenario, further highlighting the efficiency and efficacy of our model in tackling complex scheduling and routing optimization problems, even under higher redundancy conditions.

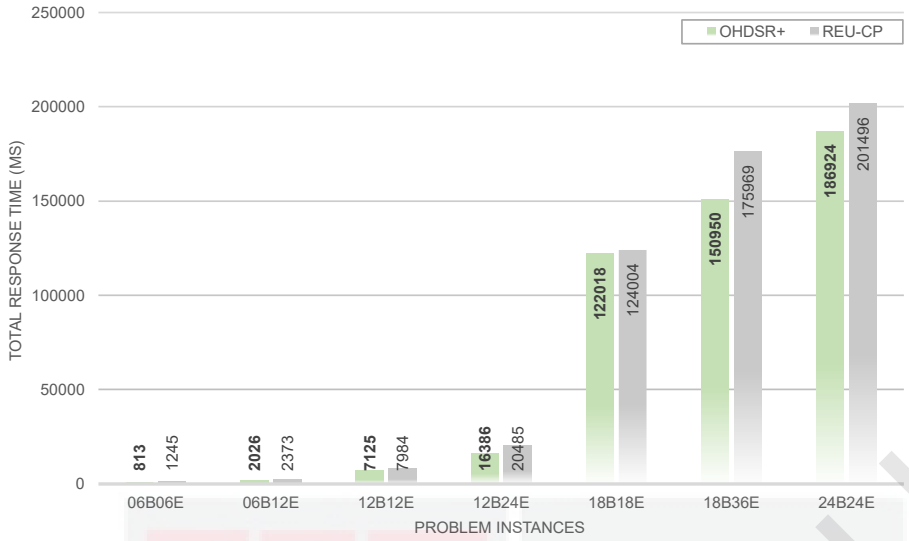

Figure 4.11 : The Total Response Time Comparison between Our Proposed OHDSR+ Model and the Related Work REU-CP [57] **Model When Applying 70% Redundancy to the Time-Triggered (TT) Streams**

While the overall performance advantage of OHDSR+ is evident, it is important to acknowledge the variations in improvement observed across different problem instances. The least significant improvement occurs in the 18B18E instance, where OHDSR+ achieves a response time reduction of only 1.60% compared to REU-CP. However, it's crucial to recognize that even this seemingly small improvement translates to a tangible reduction of 1986ms in processing time. This underscores a key point: even the minimal gains achieved by OHDSR+ represent a significant advancement over the REU-CP model, further solidifying its superior performance and efficiency.

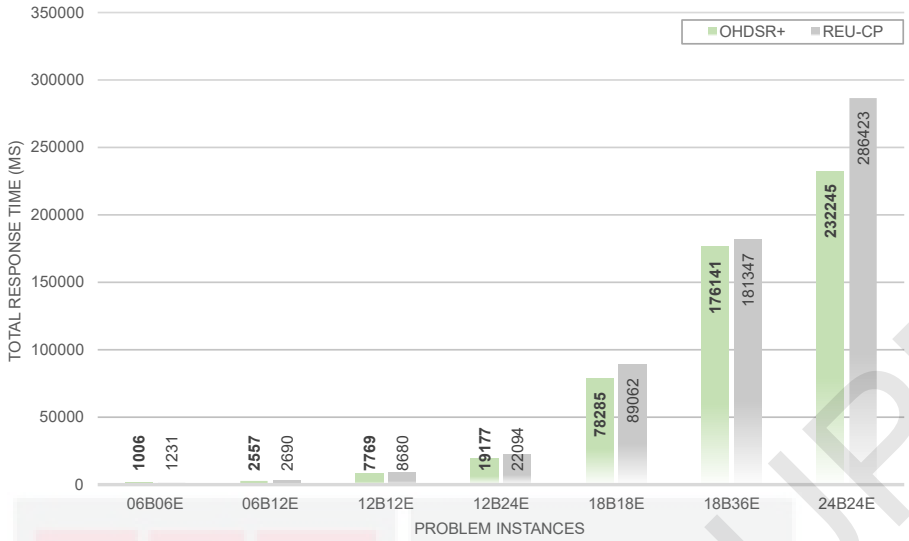

Figure 4.12 : The Total Response Time Comparison between Our Proposed OHDSR+ Model and the Related Work REU-CP [57] **Model When Applying 100% Redundancy to the Time-Triggered (TT) Streams**

The comparative analysis presented in Figure 4.12 goes beyond simply highlighting the strengths of OHDSR+ in individual scenarios. It paints a broader picture of the model's consistent efficiency when compared to the REU-CP model, particularly under the challenging conditions of 100% TT redundancy. This comprehensive comparison provides robust evidence of OHDSR+'s effectiveness in optimizing response times across a diverse range of network configurations, solidifying its position as a powerful and reliable solution for tackling complex scheduling and routing challenges in modern network environments.

To evaluate the optimization model's efficiency, the response times of its components—the scheduling and routing models—were analyzed. Focusing on the scheduling model, OHDSR+, its performance was compared to REU-CP. OHDSR+ outperformed REU-CP across all problem instances and redundancy scenarios. Figure 4.13 illustrates this advantage, showing scheduling response times for all instances with 40% TT stream redundancy. OHDSR+ achieved significant reductions, up to

63.97% in the 18B36E instance, underscoring its efficiency in time slot allocation and overall performance gains.

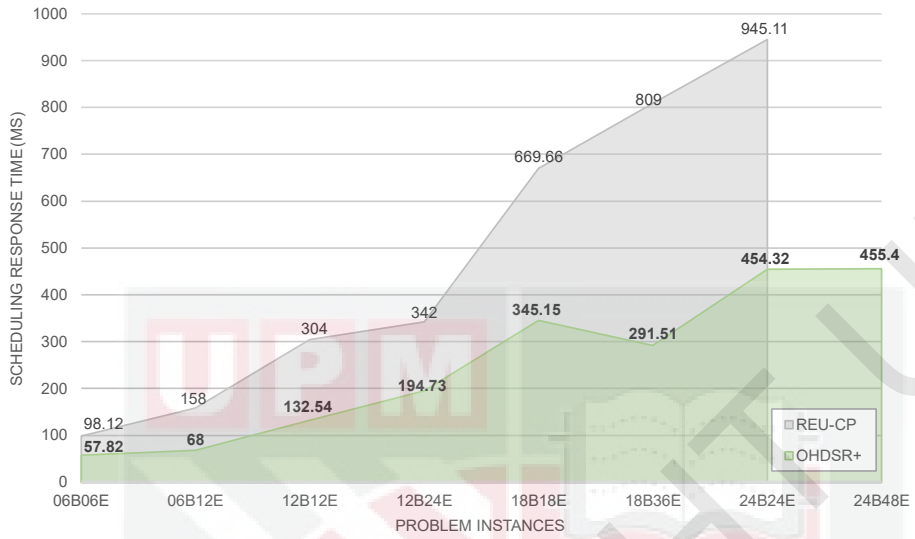

Figure 4.13 : The Scheduling Response Time Comparison between Our Proposed OHDSR+ Model and the Related Work REU-CP [57] **Model When Applying 40% Redundancy to the Time-Triggered (TT) Streams**

The performance advantage of OHDSR+ extends even to instances where the overall improvement might seem less pronounced. For example, in the 06B06E instance, while the reduction in scheduling response time is comparatively lower at 41.07%, it remains a significant improvement over the REU-CP model. This becomes particularly noteworthy when contrasted with the total response time improvement for the same redundancy scenario, which stands at a more modest 6.80%.

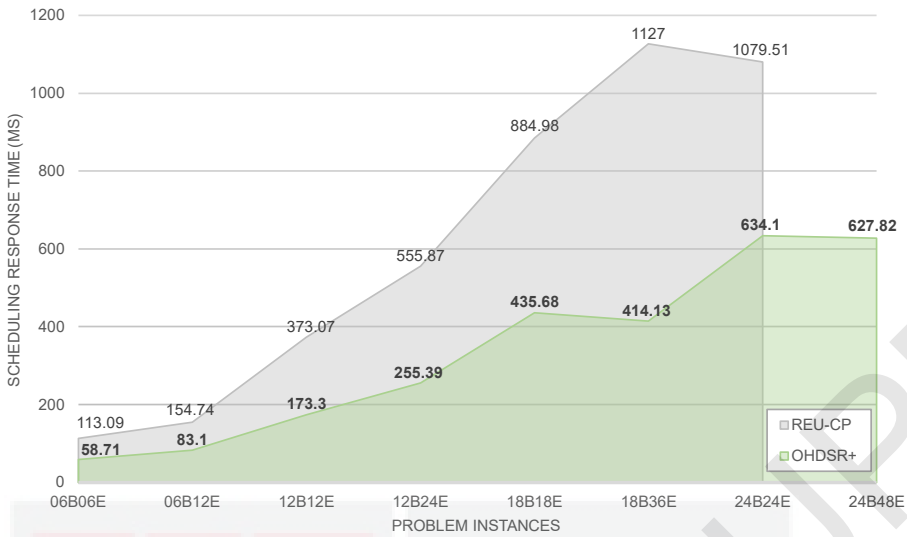

Figure 4.14 : The Scheduling Response Time Comparison between Our Proposed OHDSR+ Model and the Related Work REU-CP [57] **Model When Applying 70% Redundancy to the Time-Triggered (TT) Streams**

The efficiency of our scheduling model persists even as redundancy levels increase. Figures 4.14 and 4.15 showcase the scheduling response times for the 70% and 100% TT stream redundancy scenarios, respectively. In both cases, OHDSR+ demonstrates significantly lower scheduling response times compared to the REU-CP model. For the 70% redundancy scenario, the difference in performance between the two models varies across instances, with OHDSR+ achieving improvements ranging from a substantial 41.26% to an impressive 63.25%. This trend of superior performance continues in the 100% redundancy scenario, where OHDSR+ consistently outperforms REU-CP by a margin of 45.42% to 63.73%. These consistent results underscore the robustness and scalability of our scheduling approach, effectively handling increasingly complex scenarios with higher redundancy levels while maintaining efficiency and minimizing response times.

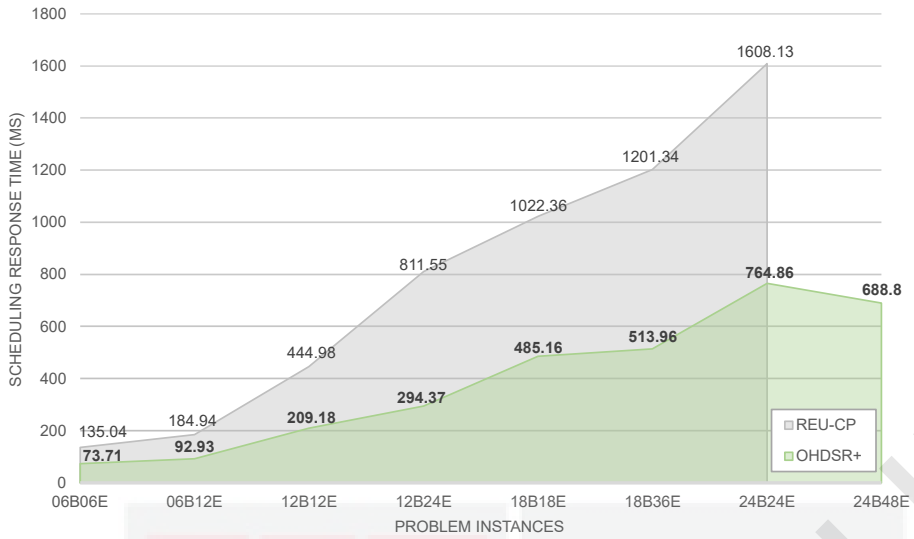

Figure 4.15 : The Scheduling Response Time Comparison between Our Proposed OHDSR+ Model and the Related Work REU-CP [57] **Model When Applying 100% Redundancy to the Time-Triggered (TT) Streams**

Among the comparisons, the case of the 24B48E instance stands out as a testament to the robustness and efficiency of OHDSR+. As evidenced in the analysis, this particular instance posed a significant challenge, with the REU-CP model failing to find a solution for its routing component. Consequently, the REU-CP model was unable to address the scheduling aspect of this instance. In contrast, OHDSR+ successfully tackled both the routing and scheduling challenges of the 24B48E instance, demonstrating its ability to handle complex scenarios with a high degree of efficiency. This outcome further reinforces the advantage of OHDSR+ in real-world applications where network configurations can be intricate and demanding.

The superior performance of the OHDSR+ model over the REU-CP model can be attributed to its innovative design features, particularly the use of a QGT for gate control lists and the incorporation of more productive constraints. The QGT provides a structured and optimized approach to managing gate control lists, enabling fine-grained control over the scheduling of TT streams. This allows OHDSR+ to handle

scheduling tasks with greater precision and adaptability, leveraging quantum-inspired computational techniques to explore multiple scheduling possibilities simultaneously. As a result, OHDSR+ achieves faster and more optimal solutions, especially in scenarios with high redundancy.

Additionally, the model incorporates productive constraints that are adaptive and reflective of real-world requirements, such as stream path reliability constraints (section 3.6.5) and stream schedule reliability constraints (section 3.6.6). These constraints enable OHDSR+ to dynamically adjust to changes in network conditions, minimizing idle times and maximizing resource efficiency. The combination of the QGT and productive constraints also enhances computational efficiency, allowing OHDSR+ to quickly evaluate and select the best scheduling options, which is critical for real-time systems requiring low latency and high reliability.

In stark contrast, the routing model exhibits significantly higher response times, as depicted in Table 4.10. This discrepancy can be attributed to the inherent complexity of routing problems. Routing involves navigating a multitude of constraints, such as link capacities, latency requirements, and traffic priorities, while simultaneously exploring a vast search space to identify the optimal path for each data stream. This intricate decision-making process naturally leads to longer processing times compared to the scheduling component.

Table 4.10 : The Routing Response Time for the Problem Instances Run on the OHDSR+ Model, When Applying 40%, 70%, and 100% Redundancy to the Time-Triggered (TT) Streams

Instance Name	Routing Response Time (ms) - OHDSR+		
	40% TT Redundancy	70% TT Redundancy	100% TT Redundancy
06B06E	577.03	666.08	825.59
06B12E	1611.10	1752.26	2168.04
12B12E	4723.99	6470.51	6945.42
12B24E	12085.00	14358.00	17004.99
18B18E	92572.80	119600.80	75341.06
18B36E	166367.72	145578.50	169897.15
24B24E	155769.25	181821.36	226220.53
24B48E	546980.60	516995.42	944054.48

Delving deeper into the response times, a stark contrast between the routing and scheduling components is uncovered. The most extreme case is observed in the 100% TT redundancy scenario for the 24B48E instance. Here, the routing model takes a staggering $944054.48ms$, equivalent to approximately $15.7minutes$, to reach a solution. This stands in stark opposition to the scheduling model's response time for the same instance and redundancy scenario, which is a mere $688.80ms$, less than a second. This vast difference in processing times underscores the inherent complexity and challenges associated with the routing model compared to its scheduling counterpart. Routing, with its intricate web of constraints and vast search space, demands significantly more computational resources and time to navigate effectively.

4.4.3 Scalability Evaluation

Scalability is paramount for TSN scheduling and routing optimization, reflecting a model's ability to maintain effectiveness with increasing problem complexity. This was evaluated by testing the OHDSR+ framework's performance on increasingly larger network topologies, expanding the network with additional devices (bridges,

end-systems) to assess the impact of network growth on solution capabilities. The scalability of the OHDSR+ model was evaluated using the diverse problem instances in Table 3.7 (varying network topology sizes, stream and task numbers), with comparative results from the REU-CP model.

The evaluation demonstrates that OHDSR+ successfully identified solutions for every instance, achieving optimal results for all eight problem instances across the three redundancy scenarios. This highlights its ability to adapt to varying network sizes and effectively manage increasing workloads. In contrast, the REU-CP model found solutions for seven out of eight problem instances in each redundancy scenario but failed to solve the most complex instance, 24B48E, which involves 24 bridges and 48 end-systems, across all three redundancy scenarios. The failure of the REU-CP model to solve the 24B48E problem instance was evident when the CP-SAT solver returned "UNKNOWN," indicating the absence of a feasible or optimal solution for that instance.

In the context of constraint programming and the CP-SAT solver [63], the status "UNKNOWN" signifies that the solver was unable to determine whether a feasible or optimal solution exists for the given problem instance within the specified time or resource limits. In the case of the REU-CP model applied to the 24B48E problem instance, the "UNKNOWN" status highlights the computational complexity of the instance, which may involve a large search space, intricate constraints, or insufficient solver configuration.

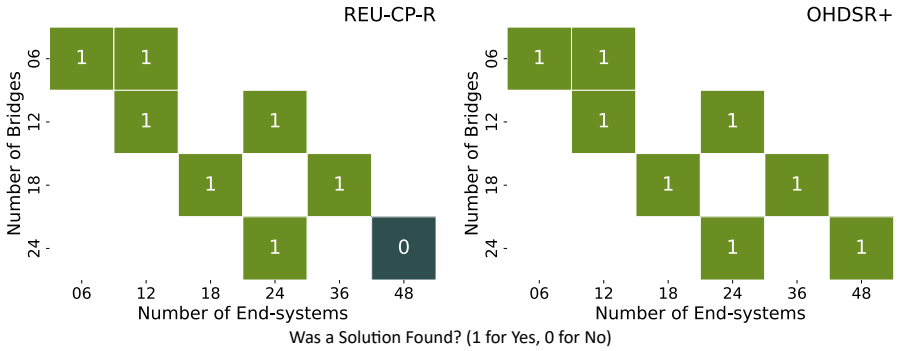

Figure 4.16 : The Scalability Comparison between Our Proposed OHDSR+ Model and the Related Work REU-CP [57] **Model**

Figure 4.16 offers a detailed visual comparison of the scalability results between the proposed OHDSR+ model and the REU-CP model. The figure employs a binary system to represent the outcomes for various network configurations, with "1" indicating that the model successfully identified a solution for a given combination of bridges (B) and end-systems (E), and "0" signifying a failure to do so. Each grid cell corresponds to a unique configuration of B and E, while empty spaces in the grid denote scenarios that were not evaluated during the testing process. Notably, the REU-CP model struggles with scalability as evidenced by the appearance of "0" in more complex network configurations, indicating its limitations in handling increasing demand. In contrast, the OHDSR+ model consistently achieves "1" across all tested configurations, demonstrating its capability to reliably find solutions even in more challenging setups. This clear distinction highlights the superior scalability and robustness of OHDSR+, making it a more suitable and dependable option for TSN applications where network configurations and demands can vary significantly.

While the OHDSR+ framework provides a comprehensive exploration of scheduling, routing, and reliability enhancement for TSN traffic, it presents some limitations. Firstly, the study exclusively focuses on TT and BE traffic, neglecting the integration

of Audio Video Bridging (AVB) traffic. AVB traffic, with its unique quality-of-service demands and bounded latency requirements, presents distinct challenges that were not addressed within this work. Secondly, the research overlooks security considerations for TT traffic, a critical aspect given the increasing interconnectedness of networks and the sensitive nature of transmitted data. The absence of secure communication protocols, encryption, and authentication mechanisms limits the applicability of the proposed solutions in environments where data confidentiality and integrity are paramount. Finally, the scheduling models assume perfect synchronization among TSN components, which may not hold true in real-world scenarios, particularly in large-scale or geographically distributed networks. This assumption overlooks the potential impact of synchronization errors on the deterministic guarantees of TSN.

4.5 Chapter Summary

This chapter presents and discusses the results of the thesis. It begins by describing the experimental setup employed in the study. To ensure the reliability of subsequent evaluations, the chapter then validates the benchmark used in the research. A substantial portion of the chapter is dedicated to evaluating the performance of the Optimized Hybrid Deterministic Scheduling and Routing (OHDSR) framework. This evaluation encompasses an examination of various problem instances, followed by in-depth assessments of both latency and response time. The chapter then proceeds to evaluate an enhanced version of the framework, known as OHDSR+. This section mirrors the structure of the OHDSR evaluation, covering problem instances, latency and response time evaluations, and additionally, a scalability evaluation. Overall, this chapter provides a comprehensive analysis of the performance of the proposed

frameworks, comparing them to existing solutions REU-CP and REU-CP. The emphasis on latency, response time, and scalability indicates a thorough examination of the frameworks' efficiency in time-sensitive networking scenarios.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The increasing demand for real-time applications across various sectors, ranging from industrial automation to autonomous vehicles, has highlighted the need for highly reliable and deterministic communication networks. Time-Sensitive Networking (TSN) emerges as a pivotal technology to address this demand, offering a suite of standards designed to achieve deterministic communication over Ethernet. However, the challenge of efficiently managing both time-triggered (TT) and best-effort (BE) traffic within TSN networks remains a complex task.

Our two models, OHDSR and OHDSR+, successfully create a robust combined scheduling and routing framework that optimizes stream paths and schedules. These frameworks demonstrate reduced latency and response times compared to contemporary state-of-the-art approaches. Additionally, both models display considerable scalability across various problem instances when compared to alternative approaches. OHDSR+ enhances the original OHDSR model by incorporating a redundancy mechanism for Time-Triggered (TT) traffic, ensuring reliable communication. Furthermore, our frameworks also account for the routing and scheduling of low-priority traffic, optimizing these elements for both Best-Effort (BE) and TT traffic.

This thesis introduces the Optimized Hybrid Deterministic Scheduling and Routing (OHDSR) approach and its enhanced version, OHDSR+, as solutions to this challenge.

Both frameworks leverage the power of constraint programming to optimize the joint scheduling and routing of TT and BE traffic, ensuring efficient utilization of network resources while meeting the stringent requirements of time-sensitive applications.

OHDSR stands out by achieving low overall delays while guaranteeing acceptable maximum delays for both TT and BE traffic. It scales well to different network sizes, outperforming similar systems in terms of speed. OHDSR+ builds on this success by adding redundant paths for real-time data, enhancing network reliability. By using a time shift mechanism, it creates a diverse timing for data transmission, further minimizing disruptions caused by potential network failures. This ensures the consistent flow of essential data, even in challenging situations.

A comprehensive evaluation of OHDSR+ demonstrates its superiority over existing approaches in terms of time efficiency, scalability, and reliability. The framework effectively reduces latency, improves response times, and efficiently optimizes even large-scale problem instances. This solidifies OHDSR+ as a valuable tool for enabling reliable and efficient communication within time-sensitive networks, paving the way for the seamless operation of real-time applications across diverse industries.

5.2 Recommendations for Future Works

This thesis delves into the realm of TSN focusing on the scheduling and routing of Time-Triggered (TT) traffic while considering the coexistence of Best-Effort (BE) traffic. Additionally, the study explores methods for enhancing reliability by implementing spatial and temporal redundancy for TT traffic, ensuring robust and dependable communication within the network. While this thesis provides a

comprehensive investigation into these aspects of TSN, several potential extensions could further enrich and broaden the scope of future research:

- Scheduling and routing of the AVB traffic: While the current thesis explores scheduling and routing strategies for Time-Triggered (TT) and Best-Effort (BE) traffic within Time-Sensitive Networking (TSN), a logical progression for future research involves integrating Audio Video Bridging (AVB) traffic into the existing framework. Characterized by specific quality of service demands and bounded latency requirements, AVB traffic presents unique challenges and opportunities for optimization within TSN. Future research endeavors could focus on developing scheduling and routing algorithms specifically tailored to the characteristics of AVB traffic. This may entail investigating methods for bandwidth reservation, delay control, and jitter minimization to ensure seamless and uninterrupted transmission of audio and video streams. Additionally, research could explore the integration of AVB-specific shaping mechanisms, such as the Credit-Based Shaper (CBS), with the existing framework to achieve a holistic solution for managing diverse traffic types within TSN. By encompassing AVB traffic, future research can contribute to the development of more versatile and comprehensive TSN solutions capable of supporting a broader spectrum of real-time applications with varying performance requirements.

- Security-Aware of the TT traffic: Although this thesis primarily investigates the enhancement of Time-Triggered (TT) traffic reliability within Time-Sensitive Networking (TSN), the criticality of security considerations warrants acknowledgment. The escalating interconnectedness of networks and the sensitive nature of transmitted data necessitate robust mechanisms to guarantee confidentiality, integrity, and authenticity. Consequently, future extensions of this research must prioritize the integration of security measures. This may involve exploring secure communication protocols within the TSN domain, such as encryption and authentication techniques, to safeguard TT traffic from unauthorized access, manipulation, or disruption. Additionally, investigating secure key management and distribution methodologies within the TSN environment is vital to ensure the overall

resilience of the security framework. By concurrently addressing reliability and security concerns, future research can facilitate a comprehensive approach to TSN that caters to both the performance and security requirements of mission-critical real-time applications.

- Synchronization-Aware scheduling: While the models presented in this thesis assume perfect synchronization among Time-Sensitive Networking (TSN) components, real-world scenarios often deviate from this ideal. Although synchronization errors may be negligible in simpler networks, such as those found in vehicular systems, they can accumulate significantly across large-scale networks with numerous nodes. These timing misalignments pose challenges for long-distance communication, potentially disrupting the deterministic guarantees that TSN aims to provide. To extend the applicability of the proposed models to more complex use cases, it is crucial to develop scheduling methods that explicitly account for synchronization discrepancies. This could involve incorporating timing margins into schedules to accommodate potential deviations or utilizing adaptive scheduling algorithms that can dynamically adjust to changing timing conditions. Additionally, exploring fault-tolerant mechanisms that can mitigate the impact of synchronization errors on critical traffic flows would further enhance the robustness of TSN systems. By addressing the challenges posed by imperfect synchronization, future research can broaden the scope of TSN and enable its application in a wider range of scenarios, from large-scale industrial automation systems to geographically distributed networks.
- Integrating Machine Learning and AI: The convergence of Machine Learning (ML) and Artificial Intelligence (AI) with Time-Sensitive Networking (TSN) presents a compelling pathway towards augmenting network intelligence, adaptability, and resilience. By harnessing the capabilities of these advanced technologies, TSN can transcend its current limitations and address emerging challenges within dynamic and complex environments.

REFERENCES

[1] "IEEE SA - The IEEE Standards Association." Accessed: Jul. 08, 2023. [Online]. Available: https://standards.ieee.org/

[2] "IEEE Standard for Ethernet," *IEEE Std 802.3-2022 (Revision of IEEE Std 802.3-2018)*, pp. 1–7025, 2022, doi: 10.1109/IEEESTD.2022.9844436.

[3] L. Zhao, P. Pop, Z. Zheng, H. Daigmorte, and M. Boyer, "Latency Analysis of Multiple Classes of AVB Traffic in TSN with Standard Credit Behavior Using Network Calculus," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 10, pp. 10291–10302, Oct. 2021, doi: 10.1109/TIE.2020.3021638.

[4] K. M. Shalghum, N. K. Noordin, A. Sali, and F. Hashim, "Worst-Case Latency Analysis for AVB Traffic Under Overlapping-Based Time-Triggered Windows in Time-Sensitive Networks," *IEEE Access*, vol. 10, pp. 43187–43208, 2022, doi: 10.1109/ACCESS.2022.3168136.

[5] T. Stuber, L. Osswald, S. Lindner, and M. Menth, "A Survey of Scheduling Algorithms for the Time-Aware Shaper in Time-Sensitive Networking (TSN)," *IEEE Access*, vol. 11, pp. 61192–61233, 2023, doi: 10.1109/ACCESS.2023.3286370.

[6] A. Nasrallah *et al.*, "Performance comparison of IEEE 802.1 TSN time aware shaper (TAS) and asynchronous traffic shaper (ATS)," *IEEE Access*, vol. 7, pp. 44165–44181, 2019, doi: 10.1109/ACCESS.2019.2908613.

[7] K. M. Shalghum, N. K. Noordin, A. Sali, and F. Hashim, "Network Calculus-Based Latency for Time-Triggered Traffic under Flexible Window-Overlapping Scheduling (FWOS) in a Time-Sensitive Network (TSN)," *Applied Sciences*, vol. 11, no. 9, p. 3896, Apr. 2021, doi: 10.3390/APP11093896.

[8] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic," *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)*, pp. 1–57, 2016, doi: 10.1109/IEEESTD.2016.8613095.

[9] "IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams," *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pp. C1-72, 2010, doi: 10.1109/IEEESTD.2009.5375704.

[10] Z. Feng, Q. Deng, M. Cai, and J. Li, "Efficient Reservation-based Fault-Tolerant Scheduling for IEEE 802.1Qbv Time-Sensitive Networking," *Journal of Systems Architecture*, vol. 123, p. 102381, Feb. 2022, doi: 10.1016/J.SYSARC.2021.102381.

[11] Y. Seol, D. Hyeon, J. Min, M. Kim, and J. Paek, "Timely Survey of Time-Sensitive Networking: Past and Future Directions," *IEEE Access*, vol. 9, pp. 142506–142527, 2021, doi: 10.1109/ACCESS.2021.3120769.

[12] A. Nasrallah *et al.*, "Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G Ull research," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 1, pp. 88–145, Jan. 2019, doi: 10.1109/COMST.2018.2869350.

[13] "IEEE 802.1 AV Bridging Task Group," IEEE Standards 802.1, Institute of Electrical and Electronics Engineers. Accessed: Jul. 13, 2023. [Online]. Available: https://www.ieee802.org/1/pages/avbridges.html

[14] S. Vitturi, C. Zunino, and T. Sauter, "Industrial Communication Systems and Their Future Challenges: Next-Generation Ethernet, IIoT, and 5G," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 944–961, Jun. 2019, doi: 10.1109/JPROC.2019.2913443.

[15] H. Zeng, C. Zhang, X. Yi, and R. Xiong, "Heuristic fragmentation-aware scheduling for a multicluster time-sensitive passive optical LAN," *Optical Fiber Technology*, vol. 66, p. 102662, Oct. 2021, doi: 10.1016/J.YOFTE.2021.102662.

[16] D. Bezerra *et al.*, "A machine learning-based optimization for end-to-end latency in TSN networks," *Comput Commun*, vol. 195, pp. 424–440, Nov. 2022, doi: 10.1016/J.COMCOM.2022.09.011.

[17] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008)*, pp. 1–499, 2020, doi: 10.1109/IEEESTD.2020.9120376.

[18] "IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications," *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pp. 1–421, 2020, doi: 10.1109/IEEESTD.2020.9121845.

[19] X. Wang, H. Yao, T. Mai, S. Guo, and Y. Liu, "Reinforcement Learning-Based Particle Swarm Optimization for End-to-End Traffic Scheduling in TSN-5G Networks," *IEEE/ACM Transactions on Networking*, vol. 31, no. 6, pp. 3254–3268, Dec. 2023, doi: 10.1109/TNET.2023.3276363.

[20] N. G. Nayak, F. Dürr, and K. Rothermel, "Routing algorithms for IEEE802.1Qbv networks," *ACM SIGBED Review*, vol. 15, no. 3, pp. 13–18, Aug. 2018, doi: 10.1145/3267419.3267421.

[21] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks – Amendment 26: Frame Preemption," *IEEE Std 802.1Qbu-2016 (Amendment to IEEE Std 802.1Q-2014)*, pp. 1–52, 2016, doi: 10.1109/IEEESTD.2016.7553415.

[22] "IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks--Amendment 29: Cyclic Queuing and Forwarding," *IEEE 802.1Qch-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd(TM)-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, IEEE Std 802.1Qbz-2016, and IEEE Std 802.1Qci-2017)*, pp. 1–30, 2017, doi: 10.1109/IEEESTD.2017.7961303.

[23] "IEEE Standard for Local and Metropolitan Area Networks--Bridges and Bridged Networks - Amendment 34: Asynchronous Traffic Shaping," *IEEE Std 802.1Qcr-2020 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018, IEEE Std 802.1Qcc-2018, IEEE Std 802.1Qcy-2019, and IEEE Std 802.1Qcx-2020)*, pp. 1–151, 2020, doi: 10.1109/IEEESTD.2020.9253013.

[24] "IEEE Standard for Local and metropolitan area networks–Frame Replication and Elimination for Reliability," *IEEE Std 802.1CB-2017*, pp. 1–102, 2017, doi: 10.1109/IEEESTD.2017.8091139.

[25] "IEEE Standard for Local and metropolitan area networks--- Bridges and Bridged Networks - Amendment 24: Path Control and Reservation," *IEEE Std 802.1Qca-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qcd-2015 and IEEE Std 802.1Q-2014/Cor 1-2015)*, pp. 1–120, 2016, doi: 10.1109/IEEESTD.2016.7434544.

[26] "IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks--Amendment 28: Per-Stream Filtering and Policing," *IEEE Std 802.1Qci-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, and IEEE Std 802.1Qbz-2016)*, pp. 1–65, 2017, doi: 10.1109/IEEESTD.2017.8064221.

[27] "IEEE Standard for Local and metropolitan area networks--Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP)," *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*, pp. 1–119, 2010, doi: 10.1109/IEEESTD.2010.5594972.

[28] "IEEE Standard for Local and Metropolitan Area Networks--Link-local Registration Protocol," *IEEE Std 802.1CS-2020*, pp. 1–151, 2021, doi: 10.1109/IEEESTD.2021.9416320.

[29] "IEEE Standard for Local and Metropolitan Area Networks--Bridges and Bridged Networks -- Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements," *IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018)*, pp. 1–208, 2018, doi: 10.1109/IEEESTD.2018.8514112.

[30] "IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks--Amendment 30: YANG Data Model," *IEEE Std 802.1Qcp-2018 (Amendment to IEEE Std 802.1Q-2018)*, pp. 1–93, 2018, doi: 10.1109/IEEESTD.2018.8467507.

[31] "IEEE Standard for Local and Metropolitan Area Networks--Bridges and Bridged Networks Amendment 33: YANG Data Model for Connectivity Fault Management," *IEEE Std 802.1Qcx-2020 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018, IEEE Std 802.1Qcc-2018, and IEEE Std 802.1Qcy-2019)*, pp. 1–123, 2020, doi: 10.1109/IEEESTD.2020.9212765.

[32] "IEEE Standard for Local and metropolitan area networks--Audio Video Bridging (AVB) Systems," *IEEE Std 802.1BA-2011*, pp. 1–45, 2011, doi: 10.1109/IEEESTD.2011.6032690.

[33] "IEEE Standard for Local and metropolitan area networks-- Audio Video Bridging (AVB) Systems-- Corrigendum 1: Technical and Editorial Corrections," *IEEE Std 802.1BA-2011/Cor 1-2016 (Corrigendum to IEEE Std 802.1BA-2011)*, pp. 1–13, 2016, doi: 10.1109/IEEESTD.2016.7520635.

[34] "IEEE Standard for Local and Metropolitan Area Networks--Audio Video Bridging (AVB) Systems," *IEEE Std 802.1BA-2021 (Revision of IEEE Std 802.1BA-2011)*, pp. 1–45, 2021, doi: 10.1109/IEEESTD.2021.9653970.

[35] "IEEE Standard for Local and metropolitan area networks -- Time-Sensitive Networking for Fronthaul," *IEEE Std 802.1CM-2018*, pp. 1–62, 2018, doi: 10.1109/IEEESTD.2018.8376066.

[36] "IEEE Standard for Local and metropolitan area networks -- Time-Sensitive Networking for Fronthaul - Amendment 1: Enhancements to Fronthaul Profiles to Support New Fronthaul Interface, Synchronization, and Syntonization Standards," *IEEE Std 802.1CMde-2020 (Amendment to IEEE Std 802.1CM-2018*, pp. 1–35, 2020, doi: 10.1109/IEEESTD.2020.9228956.

[37] L. Xu, Q. Xu, C. Chen, Y. Zhang, S. Wang, and X. Guan, "Efficient Task-Network Scheduling With Task Conflict Metric in Time-Sensitive Networking," *IEEE Trans Industr Inform*, vol. 20, pp. 1528–1538, Feb. 2023, doi: 10.1109/TII.2023.3278883.

[38] J. Min, W. Kim, and J. Paek, "Co-Optimization Framework for Heterogeneous Search Spaces in Time-Sensitive Network Planning," *IEEE Internet Things J*, vol. 11, no. 7, pp. 11779–11792, Apr. 2024, doi: 10.1109/JIOT.2023.3331430.

[39] K. M. Shalghum, N. K. Noordin, A. Sali, and F. Hashim, "Critical Offset Optimizations for Overlapping-Based Time-Triggered Windows in Time-Sensitive Network," *IEEE Access*, vol. 9, pp. 130484–130501, 2021, doi: 10.1109/ACCESS.2021.3110585.

[40] V. Gavrilut, L. Zhao, M. L. Raagaard, and P. Pop, "AVB-Aware routing and scheduling of time-triggered traffic for TSN," *IEEE Access*, vol. 6, pp. 75229–75243, 2018, doi: 10.1109/ACCESS.2018.2883644.

[41] L. Leonardi, L. Lo Bello, and G. Patti, “Bandwidth Partitioning for Time-Sensitive Networking Flows in Automotive Communications,” *IEEE Communications Letters*, vol. 25, no. 10, pp. 3258–3261, Oct. 2021, doi: 10.1109/LCOMM.2021.3103004.

[42] Y. Peng, B. Shi, T. Jiang, X. Tu, D. Xu, and K. Hua, “A Survey on In-vehicle Time Sensitive Networking,” *IEEE Internet Things J*, vol. 10, no. 16, pp. 14375–14396, Aug. 2023, doi: 10.1109/JIOT.2023.3264909.

[43] A. Arestova, K.-S. J. Hielscher, and R. German, “Optimization of Bandwidth Utilization and Gate Control List Configuration in 802.1Qbv Networks,” *IEEE Access*, vol. 11, pp. 115076–115090, 2023, doi: 10.1109/ACCESS.2023.3324957.

[44] Y. Nakayama, R. Yaegashi, A. H. N. Nguyen, and Y. Hara-Azumi, “Real-Time Reconfiguration of Time-Aware Shaper for ULL Transmission in Dynamic Conditions,” *IEEE Access*, vol. 9, pp. 115246–115255, 2021, doi: 10.1109/ACCESS.2021.3105420.

[45] P. Zhang, Y. Liu, J. Shi, Y. Huang, and Y. Zhao, “A Feasibility Analysis Framework of Time-Sensitive Networking Using Real-Time Calculus,” *IEEE Access*, vol. 7, pp. 90069–90081, 2019, doi: 10.1109/ACCESS.2019.2927516.

[46] J. Falk, F. Dürr, and K. Rothermel, “Exploring practical limitations of joint routing and scheduling for TSN with ILP,” in *Proceedings - 2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA 2018*, Institute of Electrical and Electronics Engineers Inc., Jan. 2018, pp. 136–146. doi: 10.1109/RTCSA.2018.00025.

[47] A. A. Atallah, G. B. Hamad, and O. A. Mohamed, “Routing and Scheduling of Time-Triggered Traffic in Time-Sensitive Networks,” *IEEE Trans Industr Inform*, vol. 16, no. 7, pp. 4525–4534, Jul. 2020, doi: 10.1109/TII.2019.2950887.

[48] M. Kim, D. Hyeon, and J. Paek, “eTAS: enhanced Time-Aware Shaper for Supporting Non-Isochronous Emergency Traffic in Time-Sensitive Networks,” *IEEE Internet Things J*, vol. 9, no. 13, pp. 10480–10491, Jul. 2022, doi: 10.1109/JIOT.2021.3124508.

[49] M. Vlk, Z. Hanzalek, K. Brejchova, S. Tang, S. Bhattacharjee, and S. Fu, “Enhancing Schedulability and Throughput of Time-Triggered Traffic in IEEE 802.1Qbv Time-Sensitive Networks,” *IEEE Transactions on Communications*, vol. 68, no. 11, pp. 7023–7038, Nov. 2020, doi: 10.1109/TCOMM.2020.3014105.

[50] R. S. Oliver, S. S. Craciunas, and W. Steiner, “IEEE 802.1Qbv Gate Control List Synthesis Using Array Theory Encoding,” in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Institute of Electrical and Electronics Engineers Inc., Aug. 2018, pp. 13–24. doi: 10.1109/RTAS.2018.00008.

[51] Q. Lv, X. Jiang, and X. Yang, "Making programmable packet scheduling time-sensitive with a FIFO queue," *Journal of Cloud Computing*, vol. 12, no. 1, p. 141, 2023, doi: 10.1186/s13677-023-00518-3.

[52] L. Zhao, P. Pop, and S. S. Craciunas, "Worst-Case Latency Analysis for IEEE 802.1Qbv Time Sensitive Networks Using Network Calculus," *IEEE Access*, vol. 6, pp. 41803–41815, Jul. 2018, doi: 10.1109/ACCESS.2018.2858767.

[53] K. Polachan, C. Singh, and T. V. Prabhakar, "Decentralized Dynamic Scheduling of TCPS flows and a Simulator for Time-Sensitive Networking," *ACM Transactions on Internet Technology (TOIT)*, vol. 22, no. 4, pp. 1–30, Nov. 2022, doi: 10.1145/3498729.

[54] S. Bush, "Toward Efficient Time-Sensitive Network Scheduling," *IEEE Trans Aerosp Electron Syst*, vol. 58, no. 3, pp. 1830–1842, Jun. 2022, doi: 10.1109/TAES.2021.3127311.

[55] W. Steiner, "An Evaluation of SMT-Based Schedule Synthesis for Time-Triggered Multi-hop Networks," in *2010 31st IEEE Real-Time Systems Symposium*, 2010, pp. 375–384. doi: 10.1109/RTSS.2010.25.

[56] F. Dürr and N. G. Nayak, "No-wait packet scheduling for IEEE time-sensitive networks (TSN)," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, Association for Computing Machinery, Oct. 2016, pp. 203–212. doi: 10.1145/2997465.2997494.

[57] N. Reusch, S. S. Craciunas, and P. Pop, "Dependability-aware routing and scheduling for Time-Sensitive Networking," *IET Cyber-Physical Systems: Theory & Applications*, vol. 7, no. 3, pp. 124–146, Sep. 2022, doi: 10.1049/CPS2.12030.

[58] L. de Moura, B. Dutertre, and N. Shankar, "A Tutorial on Satisfiability Modulo Theories," in *Computer Aided Verification*, W. Damm and H. Hermanns, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 20–36. doi: 10.1007/978-3-540-73368-3_5.

[59] Q.-S. Phan and P. Malacaria, "All-Solution Satisfiability Modulo Theories: Applications, Algorithms and Benchmarks," in *2015 10th International Conference on Availability, Reliability and Security*, IEEE, Aug. 2015, pp. 100–109. doi: 10.1109/ARES.2015.14.

[60] L. de Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, C. R. Ramakrishnan and J. Rehof, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 337–340. doi: 10.1007/978-3-540-78800-3_24.

[61] "IBM ILOG CPLEX Optimization Studio," Cplex, IBM ILOG. Accessed: Feb. 13, 2024. [Online]. Available: https://www.ibm.com/docs/en/icos/22.1.1

[62] "Gurobi Optimizer Reference Manual," Gurobi Optimization, LLC. Accessed: Feb. 13, 2024. [Online]. Available: https://www.gurobi.com

[63] "CP-SAT Solver | OR-Tools," Google for Developers. Accessed: Jul. 20, 2023. [Online]. Available: https://developers.google.com/optimization/cp/cp_solver

[64] H. J. Kim, K. C. Lee, and S. Lee, "A Genetic Algorithm based Scheduling Method for Automotive Ethernet," in *IECON Proceedings (Industrial Electronics Conference)*, IEEE Computer Society, Oct. 2021. doi: 10.1109/IECON48115.2021.9589998.

[65] S. D. McLean, E. A. Juul Hansen, P. Pop, and S. S. Craciunas, "Configuring ADAS Platforms for Automotive Applications Using Metaheuristics," *Front Robot AI*, vol. 8, p. 353, Jan. 2022, doi: 10.3389/FROBT.2021.762227/BIBTEX.

[66] P. Pop, M. L. Raagaard, S. S. Craciunas, and W. Steiner, "Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks," *IET Cyber-Physical Systems: Theory & Applications*, vol. 1, no. 1, pp. 86–94, Dec. 2016, doi: https://doi.org/10.1049/iet-cps.2016.0021.

[67] F. Smirnovt, M. Gla, F. Reimann, and J. Teich, "Optimizing Message Routing and Scheduling in Automotive Mixed-Criticality Time-Triggered Networks," in *Proceedings - Design Automation Conference*, Institute of Electrical and Electronics Engineers Inc., Jun. 2017. doi: 10.1145/3061639.3062298.

[68] E. Schweissguth, P. Danielis, D. Timmermann, H. Parzyjegla, and G. Mühl, "ILP-based joint routing and scheduling for time-triggered networks," in *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, New York, NY, USA: ACM, Oct. 2017, pp. 8–17. doi: 10.1145/3139258.3139289.

[69] J. Falk, F. Dürr, and K. Rothermel, "Time-Triggered Traffic Planning for Data Networks with Conflict Graphs," in *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2020, pp. 124–136. doi: 10.1109/RTAS48715.2020.00-12.

[70] J. Li, H. Xiong, Q. Li, F. Xiong, and J. Feng, "Run-Time Reconfiguration Strategy and Implementation of Time-Triggered Networks," *Electronics (Basel)*, vol. 11, no. 9, 2022, doi: 10.3390/electronics11091477.

[71] D. Hellmanns, L. Haug, M. Hildebrand, F. Dürr, S. Kehrer, and R. Hummen, "How to Optimize Joint Routing and Scheduling Models for TSN Using Integer Linear Programming," in *29th International Conference on Real-Time Networks and Systems*, New York, NY, USA: ACM, Apr. 2021, pp. 100–111. doi: 10.1145/3453417.3453421.

[72] H. Nie, S. Li, and Y. Liu, "An Enhanced Routing and Scheduling Mechanism for Time-Triggered Traffic with Large Period Differences in Time-Sensitive Networking," *Applied Sciences*, vol. 12, no. 9, 2022, doi: 10.3390/app12094448.

[73] A. Arestova, K.-S. J. Hielscher, and R. German, “Design of a Hybrid Genetic Algorithm for Time-Sensitive Networking,” in *Measurement, Modelling and Evaluation of Computing Systems*, H. Hermanns, Ed., Cham: Springer International Publishing, 2020, pp. 99–117. doi: 10.1007/978-3-030-43024-5_7.

[74] M. Nawaz, E. E. Enscore, and I. Ham, “A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem,” *Omega (Westport)*, vol. 11, no. 1, pp. 91–95, 1983, doi: https://doi.org/10.1016/0305-0483(83)90088-9.

[75] Y. Wang *et al.*, “A time-sensitive network scheduling algorithm based on improved ant colony optimization,” *Alexandria Engineering Journal*, vol. 60, no. 1, pp. 107–114, Feb. 2021, doi: 10.1016/J.AEJ.2020.06.013.

[76] M. Dorigo, M. Birattari, and T. Stutzle, “Ant colony optimization,” *IEEE Comput Intell Mag*, vol. 1, no. 4, pp. 28–39, 2006, doi: 10.1109/MCI.2006.329691.

[77] J. Min, Y. Kim, M. Kim, J. Paek, and R. Govindan, “Reinforcement learning based routing for time-aware shaper scheduling in time-sensitive networks,” *Computer Networks*, vol. 235, p. 109983, 2023, doi: https://doi.org/10.1016/j.comnet.2023.109983.

[78] Y. Wang *et al.*, “Joint Routing and GCL Scheduling Algorithm Based on Tabu Search in TSN,” in *2023 19th International Conference on Network and Service Management (CNSM)*, 2023, pp. 1–5. doi: 10.23919/CNSM59352.2023.10327834.

[79] Y. Wang, K. Yu, J. Shi, and X. Zhu, “Joint Optimization Algorithm of Traffic Scheduling and Routing of Latency Sensitive Services for Time-Sensitive Networking,” in *2023 International Conference on Networks, Communications and Intelligent Computing (NCIC)*, 2023, pp. 231–237. doi: 10.1109/NCIC61838.2023.00045.

[80] X. Liu, S. Jiang, Z. Mei, W. Hua, and H. Xu, “Joint Optimization Algorithm of Traffic Scheduling and Routing of Latency Sensitive Services for Time-Sensitive Networking,” in *2023 IEEE 3rd International Conference on Intelligent Technology and Embedded Systems (ICITES)*, 2023, pp. 128–136. doi: 10.1109/ICITES59818.2023.10356873.

[81] Z. Lin, B. Xu, and K. Qiu, “An optimization algorithm for joint routing and scheduling in time-sensitive networks,” in *2023 8th International Conference on Multimedia Communication Technologies (ICMCT)*, 2023, pp. 32–37. doi: 10.1109/ICMCT60483.2023.00013.

[82] W. Sun *et al.*, “Joint Routing and Scheduling Optimization of In-Vehicle Time-Sensitive Networks Based on Improved Grey Wolf Optimizer,” *IEEE Internet Things J*, vol. 11, no. 4, pp. 7093–7106, 2024, doi: 10.1109/JIOT.2023.3315286.

[83] S. Yang, L. Zhuang, J. Lan, J. Zhang, and B. Li, “Reuse-based online joint routing and scheduling optimization mechanism in deterministic networks,” *Computer Networks*, vol. 238, p. 110117, 2024, doi: https://doi.org/10.1016/j.comnet.2023.110117.

[84] Y. Zheng, S. Wang, S. Yin, B. Wu, and Y. Liu, “Mix-Flow Scheduling for Concurrent Multipath Transmission in Time-Sensitive Networking,” in *2021 IEEE International Conference on Communications Workshops, ICC Workshops 2021 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Jun. 2021. doi: 10.1109/ICCWORKSHOPS50388.2021.9473893.

[85] D. Yang, K. Gong, J. Ren, W. Zhang, W. Wu, and H. Zhang, “TC-Flow: Chain Flow Scheduling for Advanced Industrial Applications in Time-Sensitive Networks,” *IEEE Netw*, vol. 36, no. 2, pp. 16–24, 2022, doi: 10.1109/MNET.007.2100444.

[86] K. Gong, D. Yang, W. Zhang, and J. Ren, “An efficient scheduling approach for multi-level industrial chain flows in time-sensitive networking,” *Computer Networks*, vol. 221, p. 109516, 2023, doi: https://doi.org/10.1016/j.comnet.2022.109516.

[87] L. Xu *et al.*, “Learning-based Scalable Scheduling and Routing Co-design with Stream Similarity Partitioning for Time Sensitive Networking,” *IEEE Internet Things J*, vol. 9, no. 15, pp. 13353–13363, Aug. 2022, doi: 10.1109/JIOT.2022.3143829.

[88] X. He, X. Zhuge, F. Dang, W. Xu, and Z. Yang, “DeepScheduler: Enabling Flow-Aware Scheduling in Time-Sensitive Networking,” in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, 2023, pp. 1–10. doi: 10.1109/INFOCOM53939.2023.10228875.

[89] A. A. Syed, S. Ayaz, T. Leinmuller, and M. Chandra, “MIP-based Joint Scheduling and Routing with Load Balancing for TSN based In-vehicle Networks,” in *IEEE Vehicular Networking Conference, VNC*, IEEE Computer Society, Dec. 2020. doi: 10.1109/VNC51378.2020.9318350.

[90] A. A. Syed, S. Ayaz, T. Leinmuller, and M. Chandra, “Dynamic Scheduling and Routing for TSN based In-vehicle Networks,” in *2021 IEEE International Conference on Communications Workshops, ICC Workshops 2021 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Jun. 2021. doi: 10.1109/ICCWORKSHOPS50388.2021.9473810.

[91] S. Bhattacharjee, K. Alexandris, E. Hansen, P. Pop, and T. Bauschert, “Latency-Aware Function Placement, Routing, and Scheduling in TSN-based Industrial Networks,” in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 4248–4254. doi: 10.1109/ICC45855.2022.9839028.

[92] Y. Huang, S. Wang, T. Huang, B. Wu, Y. Wu, and Y. Liu, "Online routing and scheduling for time-sensitive networks," in *Proceedings - International Conference on Distributed Computing Systems*, Institute of Electrical and Electronics Engineers Inc., Jul. 2021, pp. 272–281. doi: 10.1109/ICDCS51616.2021.00034.

[93] R. Mahfouzi, A. Aminifar, S. Samii, A. Rezine, P. Eles, and Z. Peng, "Stability-aware integrated routing and scheduling for control applications in Ethernet networks," in *Proceedings of the 2018 Design, Automation and Test in Europe Conference and Exhibition, DATE 2018*, Institute of Electrical and Electronics Engineers Inc., Apr. 2018, pp. 682–687. doi: 10.23919/DATE.2018.8342096.

[94] R. Mahfouzi, A. Aminifar, S. Samii, A. Rezine, P. Eles, and Z. Peng, "Breaking silos to guarantee control stability with communication over ethernet TSN," *IEEE Des Test*, vol. 38, no. 5, pp. 48–56, Oct. 2021, doi: 10.1109/MDAT.2020.2968281.

[95] L. Xu, Q. Xu, Y. Zhang, J. Zhang, and C. Chen, "Co-design Approach of Scheduling and Routing in Time Sensitive Networking," in *Proceedings - 2020 IEEE Conference on Industrial Cyberphysical Systems, ICPS 2020*, Institute of Electrical and Electronics Engineers Inc., Jun. 2020, pp. 111–116. doi: 10.1109/ICPS48405.2020.9274702.

[96] Y. Zhou, S. Samii, P. Eles, and Z. Peng, "Time-Triggered Scheduling for Time-Sensitive Networking with Preemption," in *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 262–267. doi: 10.1109/ASP-DAC52403.2022.9712545.

[97] M. Vlk, Z. Hanzálek, and S. Tang, "Constraint programming approaches to joint routing and scheduling in time-sensitive networks," *Comput Ind Eng*, vol. 157, p. 107317, Jul. 2021, doi: 10.1016/J.CIE.2021.107317.

[98] E. Schweissguth, D. Timmermann, H. Parzyjegla, P. Danielis, and G. Muhl, "ILP-based routing and scheduling of multicast realtime traffic in time-sensitive networks," in *2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA 2020*, Institute of Electrical and Electronics Engineers Inc., Aug. 2020, pp. 1–11. doi: 10.1109/RTCSA50079.2020.9203662.

[99] C. Li, C. Zhang, W. Zheng, X. Wen, Z. Lu, and J. Zhao, "Joint Routing and Scheduling for Dynamic Applications in Multicast Time-Sensitive Networks," in *2021 IEEE International Conference on Communications Workshops, ICC Workshops 2021 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Jun. 2021. doi: 10.1109/ICCWORKSHOPS50388.2021.9473540.

[100] M. Pahlevan and R. Obermaisser, “Genetic Algorithm for Scheduling Time-Triggered Traffic in Time-Sensitive Networks,” in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, Institute of Electrical and Electronics Engineers Inc., Oct. 2018, pp. 337–344. doi: 10.1109/ETFA.2018.8502515.

[101] M. Pahlevan, N. Tabassam, and R. Obermaisser, “Heuristic list scheduler for time triggered traffic in time sensitive networks,” *ACM SIGBED Review*, vol. 16, no. 1, pp. 15–20, Feb. 2019, doi: 10.1145/3314206.3314208.

[102] Q. Yu, H. Wan, X. Zhao, Y. Gao, and M. Gu, “Online Scheduling for Dynamic VM Migration in Multicast Time-Sensitive Networks,” *IEEE Trans Industr Inform*, vol. 16, no. 6, pp. 3778–3788, Jun. 2020, doi: 10.1109/TII.2019.2925538.

[103] Q. Yu and M. Gu, “Adaptive Group Routing and Scheduling in Multicast Time-Sensitive Networks,” *IEEE Access*, vol. 8, pp. 37855–37865, 2020, doi: 10.1109/ACCESS.2020.2974580.

[104] H. Chahed and A. Kassler, “Optimizing TSN Routing, Scheduling, and Task Placement in Virtualized Edge-Compute Platforms,” in *2024 27th Conference on Innovation in Clouds, Internet and Networks (ICIN)*, 2024, pp. 153–157. doi: 10.1109/ICIN60470.2024.10494455.

[105] A. Alnajim, S. Salehi, and C. C. Shen, “Incremental path-selection and scheduling for time-sensitive networks,” in *2019 IEEE Global Communications Conference, GLOBECOM 2019 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Dec. 2019. doi: 10.1109/GLOBECOM38437.2019.9013427.

[106] Y. Li, J. Jiang, and S. H. Hong, “Joint Traffic Routing and Scheduling Algorithm Eliminating the Nondeterministic Interruption for TSN Networks Used in IIoT,” *IEEE Internet Things J*, vol. 9, no. 19, pp. 18663–18680, Oct. 2022, doi: 10.1109/JIOT.2022.3163411.

[107] V. Gavrilut and P. O. P. Paul, “Traffic-type Assignment for TSN-based Mixed-criticality Cyber-physical Systems,” *ACM Transactions on Cyber-Physical Systems*, vol. 4, no. 2, pp. 1–27, Jan. 2020, doi: 10.1145/337170.

[108] A. Berisa *et al.*, “AVB-aware Routing and Scheduling for Critical Traffic in Time-sensitive Networks with Preemption,” in *Proceedings of the 30th International Conference on Real-Time Networks and Systems*, in RTNS ’22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 207–218. doi: 10.1145/3534879.3534926.

[109] L. Yang, Y. Wei, F. R. Yu, and Z. Han, “Joint Routing and Scheduling Optimization in Time-Sensitive Networks Using Graph-Convolutional-Network-Based Deep Reinforcement Learning,” *IEEE Internet Things J*, vol. 9, no. 23, pp. 23981–23994, 2022, doi: 10.1109/JIOT.2022.3188826.

[110] T. Fedullo, A. Morato, F. Tramarin, L. Rovati, and S. Vitturi, "A Comprehensive Review on Time Sensitive Networks with a Special Focus on Its Applicability to Industrial Smart and Distributed Measurement Systems," *Sensors 2022, Vol. 22, Page 1638*, vol. 22, no. 4, p. 1638, Feb. 2022, doi: 10.3390/S22041638.

[111] J. Min, W. Kim, J. Paek, and R. Govindan, "Effective Routing and Scheduling Strategies for Fault-Tolerant Time-Sensitive Networking," *IEEE Internet Things J*, vol. 11, no. 6, pp. 11008–11020, Mar. 2024, doi: 10.1109/JIOT.2023.3328626.

[112] L. Lo Bello and W. Steiner, "A Perspective on IEEE Time-Sensitive Networking for Industrial Communication and Automation Systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1094–1120, Jun. 2019, doi: 10.1109/JPROC.2019.2905334.

[113] A. A. Atallah, G. B. Hamad, and O. A. Mohamed, "Fault-Resilient Topology Planning and Traffic Configuration for IEEE 802.1Qbv TSN Networks," in *2018 IEEE 24th International Symposium on On-Line Testing and Robust System Design, IOLTS 2018*, Institute of Electrical and Electronics Engineers Inc., Sep. 2018, pp. 151–156. doi: 10.1109/IOLTS.2018.8474201.

[114] F. Pozo, G. Rodriguez-Navas, and H. Hansson, "Schedule Reparability: Enhancing Time-Triggered Network Recovery Upon Link Failures," in *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2018, pp. 147–156. doi: 10.1109/RTCSA.2018.00026.

[115] A. A. Syed, S. Ayaz, T. Leinmüller, and M. Chandra, "Fault-Tolerant Dynamic Scheduling and Routing for TSN based In-vehicle Networks," in *2021 IEEE Vehicular Networking Conference (VNC)*, 2021, pp. 72–75. doi: 10.1109/VNC52810.2021.9644662.

[116] Y. Zhou, S. Samii, P. Eles, and Z. Peng, "Reliability-aware Scheduling and Routing for Messages in Time-sensitive Networking," *ACM Trans Embed Comput Syst*, vol. 20, no. 5, pp. 1–24, May 2021, doi: 10.1145/3458768.

[117] Y. Zhou, S. Samii, P. Eles, and Z. Peng, "ASIL-decomposition based routing and scheduling in safety-critical time-sensitive networking," in *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS*, Institute of Electrical and Electronics Engineers Inc., May 2021, pp. 184–195. doi: 10.1109/RTAS52030.2021.00023.

[118] "ISO 26262-1:2018 - Road vehicles: Functional safety," 2018, *ISO: the International Organization for Standardization*.

[119] V. Gavrilut, B. Zarrin, P. Pop, and S. Samii, "Fault-Tolerant Topology and Routing Synthesis for IEEE Time-Sensitive Networking," in *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, New York, NY, USA: ACM, 2017, pp. 267–276. doi: 10.1145/3139258.

[120] H. Li, H. Cheng, and L. Yang, “Reliable Routing and Scheduling in Time-Sensitive Networks,” in *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*, 2021, pp. 806–811. doi: 10.1109/MSN53354.2021.00126.

[121] L. Ji, S. He, C. Gu, Z. Shi, and J. Chen, “Routing and Scheduling for Low Latency and Reliability in Time-Sensitive Software-Defined IIoT,” *IEEE Internet Things J*, vol. 11, no. 7, pp. 12929–12940, 2024, doi: 10.1109/JIOT.2023.3337941.

[122] N. Reusch, P. Pop, and S. S. Craciunas, “Work-In-Progress: Safe and Secure Configuration Synthesis for TSN using Constraint Programming,” in *Proceedings - Real-Time Systems Symposium*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 387–390. doi: 10.1109/RTSS49844.2020.00045.

[123] Q. D. Pham and Y. Deville, “Solving the quorumcast routing problem by constraint programming,” *Constraints*, vol. 17, no. 4, pp. 409–431, 2012, doi: 10.1007/s10601-012-9125-z.

[124] S. S. Craciunas, R. Serna, O. Martin, and C. W. Steiner, “Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks,” in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, Association for Computing Machinery, Oct. 2016, pp. 183–192. doi: 10.1145/2997465.2997470.

[125] “Visual Studio Code (VS Code),” Microsoft. Accessed: May 29, 2024. [Online]. Available: https://code.visualstudio.com/docs

[126] “Python 3.11,” Python Software Foundation. Accessed: May 29, 2024. [Online]. Available: https://docs.python.org/3.11/

[127] A. A. Hagberg, D. A. Schult, and P. J. Swart, “Exploring Network Structure, Dynamics, and Function using NetworkX,” in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11–15. doi: 10.25080/TCWV9851.

[128] H. S. Thompson and C. Lilley, “XML Media Types,” Jul. 2014, *RFC Editor*. doi: 10.17487/RFC7303.

[129] “Matplotlib 3.5.3 (matplotlib.pyplot) documentation.” Accessed: Jan. 03, 2025. [Online]. Available: https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html

[130] “The ElementTree XML API (xml.etree.ElementTree) - Python 3.11.11 documentation.” Accessed: Jan. 03, 2025. [Online]. Available: https://docs.python.org/3.11/library/xml.etree.elementtree.html

[131] “Minimal DOM implementation (xml.dom.minidom) - Python 3.11.11documentation.” Accessed: Jan. 03, 2025. [Online]. Available: https://docs.python.org/3.11/library/xml.dom.minidom.html

BIODATA OF STUDENT

Bilal Omar Akram received the B.Sc. degree in computer and information engineering from the University of Mosul, Nineveh, Iraq, and the M.Sc. degree in communication engineering from Universiti Putra Malaysia, where he is currently pursuing the Ph.D. degree. His research interests include wireless communications and network systems, time-sensitive networking (TSN), constraint programming (CP) techniques, artificial intelligence (AI), machine learning, deep learning, and the optimization, performance analysis, and evaluation of real-time networks.

LIST OF PUBLICATIONS

B. O. Akram, N. K. Noordin, F. Hashim, M. F. A. Rasid, M. I. Salman and A. M. Abdulghani, "Joint Scheduling and Routing Optimization for Deterministic Hybrid Traffic in Time-Sensitive Networks Using Constraint Programming," in IEEE Access, vol. 11, pp. 142764-142779, 2023, doi: 10.1109/ACCESS.2023.3343409.

B. O. Akram, N. Kamariah Noordin, F. Hashim, M. A. Fadlee Rasid, M. Ismael Salman and A. M. Abdulghani, "Enhancing Reliability of Time-Triggered Traffic in Joint Scheduling and Routing Optimization Within Time-Sensitive Networks," in IEEE Access, vol. 12, pp. 78379-78396, 2024, doi: 10.1109/ACCESS.2024.3408923.